

FUZE BASIC

Programmer's Reference Guide



Windows™



Android



Raspberry Pi®



Linux



Programmer's Reference Guide

Introduction	3
Immediate Mode Commands	4
Functions, Constants & Procedures	12
Joystick and Gamepad commands	133
Keycodes (scanKeyboard)	134
Notes & Octaves for Sound command	135

For new commands, features and projects visit;

www.fuze.co.uk

Published in the United Kingdom ©2016 - 2016 FUZE Technologies Ltd. FUZE, FUZE BASIC, FUZE logos, designs, documentation and associated materials are Copyright FUZE Technologies Ltd.

No part of this document may be copied, reproduced and or distributed without written consent from FUZE Technologies Ltd. All rights reserved. The Arduino brand, Arduino logo and design of their boards are copyright of Arduino LLC. Any other brand names are the copyright of their respective owners.

With many thanks and a great deal of respect to Gordon Henderson for his work on RTB where FUZE BASIC originated.

FUZE BASIC is developed by FUZE Technologies Ltd in the UK. The team consists of;

Jon Silvera - Project manager
Luke Mulcahy - Lead programmer
Mike Green - Code contributor & Technical consultant
David Silvera - FUZE Product manager
Colin Bodley - Documentation
Additional information – www.fuze.co.uk

Manual Version: Date: 7th July 2016 FUZE BASIC Version: 3.6

Introduction

This reference guide aims to provide a comprehensive and detailed explanation of every command in the FUZE BASIC library.

It is an evolving document because FUZE BASIC is constantly being updated and improved. To download the latest version please visit the resources section on the www.fuze.co.uk website.

FUZE BASIC is currently available for Microsoft Windows®, Raspberry Pi, The FUZE and Linux. We hope to add Android in the coming months and possibly Mac OSx & iOS versions too.

Listed alongside each command is a series of operating system icons highlighted to show which OS the command is compatible with. For example;

ANALOGREAD

displays



The direct analog commands require the FUZE IO Board because it has a built in analog chip whereas the Raspberry Pi does not have one as standard.

SENSEPLOT

displays



..because this command will work on a standalone Pi and a FUZE with a senseHAT connected.

Installing FUZE BASIC

For details on installing and getting started we highly recommend you begin with the FUZE BASIC Project Workbook which can be downloaded for free from the FUZE website.

The website also has many other tutorials, examples and projects. You are encouraged to contribute more to help fellow coding students.

Using FUZE BASIC with an Arduino / Genuino

FUZE BASIC will only recognise an Arduino device once the driver software has been installed. Please visit the official Arduino website at <https://www.arduino.cc> and download / install the latest version.

Once installed FUZE BASIC will ask if you want to prepare it for use. It will do this with any new unit or existing ones where you might have installed a none FUZE BASIC application.

Community

We really hope you enjoy using FUZE BASIC as much as we do.

If you find yourself becoming quite proficient at FUZE BASIC then why not ~~show off~~ er.. share your work on the FUZE website. At the very least you'll be helping others to progress.

Many thanks from teamFUZE

CD	4
CHAIN	5
CLEAR	5
CLS	6
CONT	6
CONTINUE	7
DIR	7
EDIT	8
EXIT	8
LIST	8
LISTGAMEPADS	8
LOAD	9
LOOK	9
NEW	9
RUN	9
SAVE	10
TROFF	10
TRON	11
VERSION	11

CD

Purpose

Change the current directory

Syntax

CD foldername

Description

Change the current folder to the one specified.

Example

CD fuzebasic Switch to the folder named fuzebasic. This must be in the current folder to work.

CD .. Go back to the parent folder (CD ../.. will go back two folders.)

CD / Go back to root folder.

Associated

PWD, LS, LOAD, SAVE, CHAIN, DIR

CHAIN

Purpose

Load a program into memory and immediately execute it.

Syntax

CHAIN *filename*\$

Description

Loads in a program from the local non-volatile storage. As with SAVE, you need to supply the filename without any quotes. Do not include the .fuze file extension. Note, if your filename has spaces then you must enter it within quotation marks. CHAIN "my game" for example.

The program will start automatically as soon as it loads.

Example

```
REM Immediate Mode
CHAIN demos/ball
RUN
```

Associated

LOAD, SAVE, NEW

CLEAR

Purpose

Clear variable memory.

Syntax

CLEAR

Description

Clears all variables and deletes all arrays. It also removes any active sprites from the screen. Stopped programs may not be continued after a CLEAR command.

Example

```
REM Immediate Mode
VARIABLE=10
PRINT VARIABLE
CLEAR
REM Unassigned variable error!
PRINT VARIABLE
```

Associated

NEW

CLS

Purpose

Clear the video display screen.

Syntax

CLS

Description

Clears the video display screen and places the cursor in the top left corner. The background is set to the current background (PAPER) colour.

Example

```
PAPER=WHITE
INK=BLACK
CLS
PRINT "Hello World"
END
```

Associated

[CLS2](#), [INK](#), [PAPER](#), [UPDATE](#)

CONT

Purpose

Continue running a program that has been stopped.

Syntax

CONT

Description

Continues program execution after a STOP instruction.

Variables are not cleared.

Example

```
REM Immediate mode
CONT
```

Associated

[STOP](#)

CONTINUE

Purpose

Continue a loop.

Syntax

CONTINUE

Description

Will cause the loop to re-start at the line containing the LOOP instruction, continuing a FOR instruction and re-evaluating any WHILE or UNTIL instructions.

Example

```
REM Prints 1 to 10 but skips over 5
FOR I=1 to 10 LOOP
  IF (I=5) THEN CONTINUE
  PRINT I
REPEAT
END
```

Associated

BREAK, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT
UNTIL, UNTIL REPEAT, WHILE REPEAT

DIR

Purpose

List FUZE BASIC program files in the current or specified directory.

Syntax

DIR [*directory*]

Description

Lists the program files in your current working directory or the optional specified *directory*. These will have the .fuz file extension.

Example

```
REM Immediate Mode
DIR
```

Associated

LOAD, SAVE

EDIT

Purpose

Edit the program in memory.

Syntax

EDIT

Description

Edits the program in memory using a full screen editor.

Example

REM Immediate mode
EDIT



EXIT

Purpose

Exit FUZE BASIC and return to the environment.

Syntax

EXIT

Description

Exit FUZE BASIC and return to the environment you started it in.

Example

REM Immediate mode
EXIT



LIST

Purpose

List the program stored in memory to the screen.

Syntax

LIST

Description

This lists the program stored in memory to the screen. You can pause the listing with the space-bar and terminate it with the ESC key.



LISTGAMEPADS

Purpose

To display any connected joysticks and gamepads.

Syntax

LISTGAMEPADS

Description

This lists, by name, any attached joystick devices that are currently connected via USB or wireless USB, if supported by the OS.

Example

REM Immediate Mode
LISTGAMEPADS



LOAD

Purpose

Load a program into memory.

Syntax

LOAD *filename*\$

Description

Loads in a program from the local non-volatile storage. As with SAVE, you need to supply the filename without any quotes. Do not include the .fuze file extension. Note, if your filename has spaces then you must enter it within quotation marks. LOAD "my game" for example.

Example

```
REM Immediate Mode
LOAD demos/ball
RUN
```

Associated

SAVE

LOOK

Purpose

Not much at all

Syntax

LOOK

Description

Erm...



NEW

Purpose

Start a new program

Syntax

NEW

Description

Deletes the program in memory. There is no verification and once it's gone, it's gone. Remember to save first!

Associated

CLEAR



RUN

Purpose

Runs the program in memory.

Syntax

RUN

Description

Runs the program in memory. Note that using RUN will clear all variables.

Example

```
REM Immediate Mode
RUN
```



SAVE

Purpose

Saves your program to the local non-volatile storage.

Syntax

SAVE *filename*\$

Description

Saves your program to the local non-volatile storage. The *filename*\$ is the name of the file you wish to save and may not contain spaces. If you have already saved a file, then you can subsequently execute SAVE without the filename and it will overwrite the last file saved. (This is reset when you load a new program or use the NEW command)

Example

```
REM Immediate Mode  
SAVE testprog
```

Associated

LOAD

TROFF

Purpose

Disables TRON

Syntax

TROFF

Description

If TRON is enabled then TROFF switches it off.

Example

```
REM Immediate Mode  
TROFF
```

Associated

TRON

TRON

Purpose

Enables TRON mode for debugging.

Syntax

TRON

Description

Switching TRON on has a very dramatic effect. Running your program when TRON is enabled displays a section of code on screen to show exactly which line is being executed.

You can adjust TRON settings with the following controls;

CTRL+ UP / DOWN to change speed

CTRL+ LEFT to Pause (hold down)

CTRL+ LEFT + RIGHT to step

CTRL+ RIGHT toggle transparent

CTRL+ SHIFT + UP toggle TRON display

CTRL+ PAGEUP increase number of lines displayed

CTRL+ PAGEDOWN decrease number of lines displayed

CTRL+ SHIFT + PAGEUP move TRON display up

CTRL+ SHIFT + PAGEDOWN move TRON display down

Example

```
REM Immediate Mode
```

```
TROFF
```

Associated

TRON



VERSION

Purpose

Print the current version of FUZE BASIC.

Syntax

VERSION

Description

Print the current version of FUZE BASIC.

Example

```
REM Immediate Mode
```

```
VERSION
```



Functions, Constants & Procedures

[main index](#)

ABS	14	COLOUR	28
ACOS	14	COPYREGION	29
ADVANCESPRITE	15	COS	30
ANALOGREAD	15	LOOP	30
ANALOGWRITE	16	LOOP REPEAT	31
ARDUINODIGITALREAD	16	DATA	31
ARDUINODIGITALWRITE	17	DATE\$	32
ARDUINOPINMODE	17	DEF FN	33
ARDUINOANALOGWRITE	18	DEF PROC	33
ARDUINOANALOGREAD	18	DEFCHAR	34
ARMBODY	19	DEG	34
ARMELBOW	19	DIGITALREAD	35
ARMGRIPPER	20	DIGITALWRITE	35
ARMLIGHT	20	DIM	36
ARMRESET	21	DRCANALOGREAD	37
ARMSHOULDER	21	DRCCLOSE	37
ARMWRIST	22	DRCDIGITALREAD	38
ASC	22	DRCDIGITALWRITE	38
ASIN	23	DRCPWMWRITE	39
ATAN	23	DRCPWMWRITE	39
BREAK	24	ELLIPSE	40
CHR\$	24	ELSE	41
CIRCLE	25	END	41
CLEARKEYBOARD	25	ENDIF	42
CLOCK	26	ENDPROC	42
CLONESPRITE	26	ENVELOPE	43
CLOSE	27	EOF	44
CLS	27	EXP	44
CLS2	28		

FADEOFF	45	INKEY	60
FADEON	45	INPUT	61
FADING	46	INPUT#	61
FALSE	46	INT	62
FFWD	47	LEFT	62
FN	47	LEFT\$	63
FONTSIZE	48	LEN	63
FOR LOOP	48	LINE	64
FREEIMAGE	49	LINETO	64
FULLSCREEN	49	LOADIMAGE	65
GET	50	LOADMUSIC	65
GET\$	50	LOADSAMPLE	66
GETIMAGEH	51	LOADSPRITE	66
GETIMAGEW	51	LOCAL	67
GETMOUSE	52	LOG	67
GETPIXEL	52	MAX	68
GETPIXELRGB	53	MICROTIME	68
GETSPRITEANGLE	53	MID\$	69
GETSPRITEH	54	MIN	69
GETSPRITEW	54	MODE7PLOTIMAGE	70
GETSPRITEX	55	MODE7PLOTSPRITE	70
GETSPRITEY	55	MOUSEOFF	71
GHEIGHT	56	MOUSEON	71
GRABREGION	56	MOUSEX	72
GWIDTH	57	MOUSEY	72
HIDESPRITE	57	MOVE	73
HLINE	58	MOVETO	73
HTAB	58	NEWSPRITE	74
HVTAB	59	NUMFORMAT	75
IF THEN	59	OPEN	75
INK	60	ORIGIN	76

Functions, Constants & Procedures

[main index](#)

PAPER	76	REWIND	91	SENSELINE	103	SPUT\$	120
PAPEROFF	77	RGBCOLOUR	92	SENSEPLOT	103	SQRT	120
PAPERON	77	RIGHT	92	SENSEPRESSURE	104	SREADY	121
PAUSECHAN	78	RIGHT\$	93	SENSERECT	104	STOP	121
PAUSEMUSIC	78	ROTATEIMAGE	93	SENSERGBCOLOUR	105	STOPCHAN	122
PENDOWN	79	RND	94	SENSESCROLL	105	STOPMUSIC	122
PENUP	79	SAVEREGION	94	SENSETEMPERATURE	106	STR\$	123
PI	80	SAVESCREEN	95	SENSEVFLIP	106	SWAP	123
PI2	80	SCALEIMAGE	95	SETCHANVOL	107	SWITCH	124
PINMODE	80	SCANKEYBOARD	96	SETMODE	107	TAN	125
PLAYMUSIC	81	SCLOSE	96	SETMOUSE	108	TANGLE	125
PLAYSAMPLE	81	SCROLLDOWN	97	SETMUSICVOL	108	THEIGHT	126
PLOT	82	SCROLLLEFT	97	SETSPRITEALPHA	109	TIME	126
PLOTIMAGE	82	SCROLLRIGHT	97	SETSPRITEANGLE	109	TIME\$	127
PLOTSPRITE	83	SCROLLUP	97	SETSPRITEFLIP	110	TONE	127
POLYEND	83	SEED	98	SETSPRITEORIGIN	110	TRIANGLE	128
POLYPLOT	84	SEEK	98	SETSPRITESIZE	111	TRUE	128
POLYSTART	84	SENSEACCELX	99	SETSPRITETRANS	111	TWIDTH	129
PLOTTEXT	85	SENSEACCELY	99	SGET	112	UPDATEMODE	129
PRINT	85	SENSEACCELZ	99	SGET\$	113	UNTIL REPEAT	130
PRINT#	86	SENSECLS	99	SGN	113	UPDATE	130
PRINTAT	86	SENSECOMPASSX	100	SHOWKEYS	114	VAL	131
PROC	87	SENSECOMPASSY	100	SIN	114	VLINE	131
PWMWRITE	87	SENSECOMPASSZ	100	SOFTPWMWRITE	115	VTAB	132
RAD	88	SENSEGETRGB	100	SOPEN	115	WAIT	132
READ	88	SENSEGYROX	101	SOUND	116	WHILE REPEAT	133
RECT	89	SENSEGYROY	101	SPACE\$	116		
REPEAT UNTIL	89	SENSEGYROZ	101	SPRITECOLLIDE	117		
RESTORE	90	SENSEHEIGHT	101	SPRITECOLLIDEP	118		
RESUMECHAN	90	SENSEHFLIP	102	SPRITEOUT	119		
RESUMEMUSIC	91	SENSEHUMIDITY	102	SPUT	119		

ABS

Purpose

Return the absolute value of the argument.

Syntax

positivenumber=ABS(*number*)

Description

Returns the absolute value of the supplied argument *number* i.e. If the argument is negative, make it positive.

Example

```
PRINT FN ElapsedYears(1966,2013)
PRINT FN ElapsedYears(2013,1966)
END
REM Return Number of years elapsed
REM Between two dates
DEF FN ElapsedYears(Year1, Year2)
=ABS(Year1-Year2)
```

Associated

SGN

ACOS

Purpose

Returns the arc cosine of the supplied argument.

Syntax

angle=ACOS(*cosine*)

Description

This is the inverse of the COS function returning the angle for a given cosine.

Example

```
PRINT "Angle with cosine = 0.5: "
DEG
REM 60 Degrees
PRINT "In Degrees: ";ACOS(0.5)
RAD
REM PI/3 Radians
PRINT "In Radians: "; ACOS(0.5)
CLOCK
PRINT "In Minutes: "; ACOS(0.5)
END
```

Associated

ATAN, COS, SIN, TAN

ADVANCESPRITE

Purpose

Advances a sprite a specified amount

Syntax

ADVANCESPRITE(*sprite*, *distance*)

Description

Moves a sprite forward by the specified distance. The direction is set by the SETSPRITEANGLE function. This is very useful when used with rotating sprites.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE("logo.bmp", pic,0)
PLOTSPRITE( pic, gWidth / 2, gHeight / 2, 0 )
FOR angle = 0 TO 360 LOOP
SETSPRITEANGLE( pic, angle )
ADVANCESPRITE( pic, 5 )
UPDATE
REPEAT
END
```

Associated

[PLOTSPRITE](#), [SETSPRITEANGLE](#)

ANALOGREAD

Purpose

Returns the value from an analog input.

Syntax

ANALOGREAD(0)

Description

Returns the a value of between 0 and 255 (0V & 3.3V) from the specified analog pin. There are four inputs, 0 to 3.

Example

```
REM Attach an LDR to 3.3V and analog pin 0
PRINT analogRead(0)
REM cover the LDR and run again.
END
```

Associated

[ANALOGREAD](#), [ANALOGWRITE](#), [ARDUINOANALOGREAD](#), [ARDUINOANALOGREAD](#)

ANALOGWRITE

Purpose

Sends a value to the analog output.

Syntax

ANALOGWRITE(0,value)

Description

Sends a voltage between 0V and 3.3V (as a value of 0 to 255) to analog pin 0. There is one analog output.

Example

REM Attach an LED to analog OUT pin 0 (the longer leg) and the shorter leg to GND

LOOP

FOR volt= 0 TO 255 LOOP

analogWrite(0,volt)

REPEAT

REPEAT

END

Associated

ANALOGREAD, ANALOGWRITE, ARDUINOANALOGREAD,
ARDUINOANALOGWRITE

ARDUNIODIGITALREAD

Purpose

Returns the value from a digital input from an Arduino compatible device

Syntax

ARDUINODIGITALGREAD(2)

ADNALOGREAD(2)

Description

Returns a value of 1 or 0 from the specified digital pin depending on the the signal being hi or low.

Example

CLS

APINMODE (2, 1)

ADWRITE (2, NOT ADREAD (2))

PRINT ADREAD (2)

END

Associated

DIGITALREAD, DIGITALWRITE, ARDUINODIGITALWRITE

ARDUNIODIGITALWRITE

Purpose

Set the state of a digital pin on an Arduino compatible device.

Syntax

```
ARDUNIODIGITALWRITE(pinno,pinvalue)  
ADWRITE(pinno,pinvalue)
```

Description

This procedure sets a digital pin to the supplied value - 0 for off or 1 for on. As with DigitalRead, you may need to set the pin mode (to output) beforehand.

Example

```
CLS  
APINMODE (2, 1)  
ADWRITE (2, NOT ADREAD (2))  
PRINT ADREAD (2)  
END
```

Associated

[DIGITALREAD](#), [DIGITALWRITE](#), [ARDUNIODIGITALREAD](#)

ARDUINOPINMODE

Purpose

Configure the mode of a digital pin on an Arduino compatible device.

Syntax

```
ARDUINOPINMODE(pinno,pinmode)  
APINMODE(pinno,pinmode)
```

Description

Configures the mode of a pin on the Pi's GPIO. It takes an argument which specifies the mode of the pin - input, output or PWM output. The modes are:

0 pinINPUT
1 pinOUTPUT

Example

```
CLS  
APINMODE (2, 1)  
ADWRITE (2, NOT ADREAD (2))  
PRINT ADREAD (2)  
END
```

Associated

[PINMODE](#), [SOFTPWMMODE](#), [ARDUINOPINMODE](#)

ARDUNIOANALOGREAD

Purpose

Returns the value from an analog input on an Arduino compatible device.

Syntax

```
ARDUINOANALOGREAD( 0 )
AAREAD( 0 )
```

Description

Returns a value between 0 and 1023 depending on the current attached to an analog pin on an Arduino compatible device.

Example

REM Attach an LDR (leg 1) to GND and (leg 2) to analog pin 0 and then a 10k resistor from (leg 2) to 5V.

```
LOOP
PRINTAT ( 0, 0 ); AAREAD(0); "    "
REPEAT
END
```

Associated

[DIGITALREAD](#), [DIGITALWRITE](#), [ARDUINODIGITALWRITE](#)

ARDUNIOANALOGWRITE

Purpose

Write a value to an analog output on an Arduino compatible device.

Syntax

```
ARDUINOANALOGWRITE( 0 )
AAWRITE( 0 )
```

Description

Send a value between 0 and 1023 depending on the output required.

Example

REM Attach an LED (5V) to digital pin 11 and GND on an Arduino compatible device.

```
CLS
APINMODE (11, 1)
LOOP
FOR v=0 TO 1023 LOOP
AAWRITE (11,v)
REPEAT
REPEAT
END
```

Associated

[ANALOGREAD](#), [ANALOGWRITE](#), [ARDUINOANALOGREAD](#),

ARMBODY

Purpose

Move the robot arm body.

Syntax

ARMBODY(*direction*)

Description

Activates the motor in the base of the robot arm according to the *direction* argument as follows:

1 Move clockwise

0 Stop

-1 Move anti-clockwise.

Example

REM Move the body for 2 seconds clockwise

ARMBODY(1)

WAIT(2)

ARMBODY(0)

END

Associated

ARMELBOW, ARMGRIPPER, ARMLIGHT, ARMRESET,
ARMSHOULDER, ARMWRIST

ARMELBOW

Purpose

Move the robot arm elbow.

Syntax

ARMELBOW(*direction*)

Description

Activates the motor in the elbow of the robot arm according to the *direction* argument as follows:

1 Move clockwise

0 Stop

-1 Move anti-clockwise.

Example

REM Move the elbow for 1s anti-clockwise

ARMELBOW(-1)

WAIT(1)

ARMELBOW(0)

END

Associated

ARMBODY, ARMGRIPPER, ARMLIGHT, ARMRESET,
ARMSHOULDER, ARMWRIST

ARMGRIPPER

Purpose

Open or close the robot arm gripper.

Syntax

ARMGRIPPER(*direction*)

Description

Activates the motor in the gripper of the robot arm according to the *direction* argument as follows:

- 1** Open gripper
- 0** Stop
- 1** Close gripper

Example

```
REM Open the gripper for 1 seconds
ARMGRIPPER(1)
WAIT(1)
ARMGRIPPER(0)
END
```

Associated

ARMBODY, ARMELBOW, ARMLIGHT, ARMRESET,
ARMSHOULDER, ARMWRIST

ARMLIGHT

Purpose

Switch the robot's LED on or off.

Syntax

ARMLIGHT(*switch*)

Description

If the *switch* argument is **1** then the LED is illuminated and if it is **0** it is switched off.

Example

```
REM Flash the LED with 1 second interval
LOOP
    ARMLIGHT(1)
    WAIT(1)
    ARMLIGHT(0)
    WAIT(1)
REPEAT
```

Associated

ARMBODY, ARMELBOW, ARMGRIPPER, ARMRESET,
ARMSHOULDER, ARMWRIST

ARMRESET

Purpose

Reset the robot arm.

Syntax

ARMRESET

Description

Stops any moving motors on the robot arm and switches off the LED.

Example

```
ARMLIGHT(1)
ARMELBOW(1)
WAIT(2)
ARMRESET
END
```

Associated

[ARMBODY](#), [ARMELBOW](#), [ARMGRIPPER](#), [ARMLIGHT](#),
[ARMSHOULDER](#), [ARMWRIST](#)

ARMSHOULDER

Purpose

Move the robot arm shoulder.

Syntax

ARMSHOULDER(*direction*)

Description

Activates the motor in the shoulder of the robot arm according to the *direction* argument as follows:

1 Move clockwise

0 Stop

-1 Move anti-clockwise.

Example

```
REM Move shoulder for 1s clockwise & back
ARMSHOULDER(1)
WAIT(1)
ARMSHOULDER(-1)
WAIT(1)
ARMSHOULDER(0)
END
```

Associated

[ARMBODY](#), [ARMELBOW](#), [ARMGRIPPER](#), [ARMLIGHT](#),
[ARMRESET](#), [ARMWRIST](#)

ARMWRIST

Purpose

Move the robot arm wrist.

Syntax

ARMWRIST(*direction*)

Description

Activates the motor in the wrist of the robot arm according to the *direction* argument as follows:

1 Move clockwise

0 Stop

-1 Move anti-clockwise.

Example

REM Move wrist for 1s clockwise & back

ARMWRIST(1)

WAIT(1)

ARMWRIST(-1)

WAIT(1)

ARMWRIST(0)

END

Associated

ARMBODY, ARMELBOW, ARMGRIPPER, ARMLIGHT,
ARMRESET, ARMSHOULDER

ASC

Purpose

Return the ASCII code for a string character.

Syntax

asciicode=ASC(*string*)

Description

Returns the ASCII value represented by the first character of *string* e.g. "A" would return 65. It is the opposite of the CHR\$ function.

Example

code=ASC("A")

PRINT "ASCII value of letter A is: "; code

PRINT "ASCII char of code 65: "; CHR\$(code)

END

Associated

CHR\$

ASIN

Purpose

Returns the arc sine of the supplied argument.

Syntax

angle = ASIN(*sine*)

Description

This is the inverse of the SIN function returning the angle for a given sine.

Example

```
PRINT "Angle with sine = 0.5: "  
DEG  
REM 30 Degrees  
PRINT "In Degrees: ";ASIN(0.5)  
RAD  
REM PI/6 Radians  
PRINT "In Radians: ";ASIN(0.5)  
CLOCK  
PRINT "In Minutes: ";ASIN(0.5)  
END
```

Associated

ACOS, ATAN, COS, SIN, TAN

ATAN

Purpose

Returns the arc tangent of the supplied argument.

Syntax

angle=ATN(*tangent*)

Description

This is the inverse of the TAN function returning the angle for a given tangent.

Example

```
PRINT "Angle with tangent equal to 1: "  
DEG  
REM 45 Degrees  
PRINT "In Degrees: ";ATAN(1)  
RAD  
REM PI/4 Radians  
PRINT "In Radians: ";ATAN(1)  
CLOCK  
PRINT "In Minutes: ";ATAN(1)  
END
```

Associated

ACOS, ASIN, COS, SIN, TAN

BREAK

Purpose

Provide an early exit from a loop.

Syntax

BREAK

Description

Terminates a loop before the terminating condition is met.

Example

```
REM Loop until the space bar is pressed
PRINT "Press the space bar to continue"
LOOP
    IF INKEY=32 THEN BREAK
REPEAT
END
```

Associated

CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT
UNTIL, UNTIL REPEAT, WHILE REPEAT

CHR\$

Purpose

Returns the string character for the specified ASCII code.

Syntax

character\$ = CHR\$(*ascii*code)

Description

Returns a one-character string consisting of the character corresponding to the ASCII code indicated by the *ascii*code argument. This is the opposite of the ASC function.

Example

```
PRINT "The following print the letter A"
PRINT CHR$(65)
PRINT CHR$(ASC("A"))
PRINT "The following print the number 3"
PRINT CHR$(51)
PRINT CHR$(ASC("0") + 3)
END
```

Associated

ASC

CIRCLE

Purpose

Draw a circle on the screen.

Syntax

CIRCLE(*centrex*,*centrey*,*radius*,*fill*)

Description

Draws a circle at position (*centrex*,*centrey*) with the specified *radius* in the current foreground COLOUR. The final parameter *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Example

```
CLS
PRINT "Draw a filled yellow circle ";
PRINT "In the centre of the screen"
COLOUR=yellow
CIRCLE(GWIDTH/2,GHEIGHT/2,100,TRUE)
UPDATE
END
```

Associated

ELLIPSE, RECT, TRIANGLE

CLEARKEYBOARD

Purpose

Clear all pending keyboard input.

Syntax

CLEARKEYBOARD

Description

Clears the keyboard input buffer.

Example

```
PRINT "Press space to continue"
WHILE NOT SCANKEYBOARD(scanSpace) LOOP
REPEAT
CLEARKEYBOARD
END
```

Associated

SCANKEYBOARD

CLOCK

Purpose

Set angle units to minutes.

Syntax

CLOCK

Description

Switches the internal angle system to clock mode. There are 60 minutes in a full circle.

Example

```
PRINT "ATAN(1) = "  
DEG  
PRINT ATAN(1); " Degrees"  
CLOCK  
PRINT ATAN(1); " Minutes"  
RAD  
PRINT ATAN(1); " Radians"  
END
```

Associated

DEG, RAD

CLONESPRITE

Purpose

Create a new sprite from an existing one.

Syntax

CLONESPRITE (*sprite handle*)

Description

Copies a sprite and its settings from an existing one (*sprite handle*) into a new one.

Example

```
sprite = newSprite (1)  
loadSprite ("ballBlue.png", sprite, 0)  
sprite2 = cloneSprite (sprite)  
LOOP  
    PlotSprite (sprite, 100, 100, 0)  
    PlotSprite (sprite2, 200, 200, 0)  
REPEAT
```

Associated

newSprite, loadSprite

CLOSE

Purpose

Close a file after use.

Syntax

`CLOSE(handle)`

Description

The CLOSE instruction closes the file specified by *handle* after use and makes sure all data is securely written to the storage medium.

Example

```
handle = OPEN("testfile.txt")
PRINT# handle, "Colin"
CLOSE(handle)
handle = OPEN("testfile.txt")
INPUT# handle, Name$
CLOSE(handle)
PRINT "Name: " + Name$
END
```

Associated

[EOF](#), [FWD](#), [INPUT#](#), [OPEN](#), [PRINT#](#), [REWIND](#), [SEEK](#)

CLS

Purpose

Clear screen.

Syntax

`CLS`

Description

Completely clear the screen of text and graphics. Note, this does not clear sprites from the screen.

Example

```
PRINT "Hello World"
WAIT (1)
CLS
END
```

Associated

[CLS](#), [UPDATE](#)

CLS2

Purpose

Clear screen without an update.

Syntax

CLS2

Description

Ideally suited to games and graphical programming. CLS2 clears the background or buffer screen and does not issue an update command. This means you can wipe the screen buffer, redraw on it and then issue an update. This ensures flicker free updates. It is also much faster than CLS.

Example

```
y = GHEIGHT / 2
radius = GWIDTH / 10
FOR x = 0 TO GWIDTH STEP radius LOOP
CLS
    COLOUR = PINK
    CIRCLE (x, y, radius, 1)
    UPDATE
REPEAT
FOR x = GWIDTH TO 0 STEP -10 LOOP
    CLS2
    COLOUR = YELLOW
    CIRCLE (x, y, radius, 1)
    UPDATE
REPEAT
```

Associated

CLS, UPDATE



COLOUR

Purpose

Set/Read the current graphics plot colour.

Syntax

COLOUR=*setcolour*

Description

Set/Read the current graphics plot colour as follows:

0 Black	10 Pink	20 Teal
1 DarkGrey	11 LightPink	21 Cyan
2 Grey	12 DarkGreen	22 Aqua
3 Silver	13 Green	23 LightBlue
4 LightGrey	14 BrightGreen	24 Brown
5 White	15 Olive	25 LightBrown
6 Maroon	16 Lime	26 Orange
7 Red	17 LightGreen	27 Gold
8 Purple	18 Navy	28 Yellow
9 Raspberry	19 Blue	29 LightYellow



Example

CLS

FOR c=0 TO 29 LOOP

COLOUR=c

CIRCLE(50,GHEIGHT-c*40-50,20,TRUE)

COLOUR=WHITE

CIRCLE(50,GHEIGHT-c*40-50,20,FALSE)

READ NameOfColour\$

HVTAB(5,c*2+2)

PRINT NameOfColour\$;" ";c

REPEAT

UPDATE

DATA "Black", "DarkGrey", "Grey", "Silver"

DATA "LightGrey", "White", "Maroon", "Red"

DATA "Purple", "Raspberry", "Pink", "LightPink"

DATA "DarkGreen", "Green", "BrightGreen", "Olive"

DATA "Lime", "LightGreen", "Navy", "Blue", "Teal"

DATA "Cyan", "Aqua", "LightBlue", "Brown"

DATA "LightBrown", "Orange", "Gold", "Yellow"

DATA "LightYellow"

END

Associated[INK](#), [PAPER](#), [RGBCOLOUR](#)

COPYREGION

Purpose

Copy a region of the screen from one location to another.

SyntaxCOPYREGION(*oldx,oldy,width,height,newX,newY*)**Description**

This enables you to duplicate the contents of a section of the screen from one place to another. This could be used for example to create a background for a game by drawing an image and then duplicating it. The region to be copied is specified by the rectangle at coordinates (*oldx,oldy*) *width* pixels wide and *height* pixels high. The region is copied to coordinates (*newx,newy*)

Example

COLOUR=RED

RECT(0,0,50,50,TRUE)

COLOUR=WHITE

LINE(0,50,50,50)

LINE(0,25,50,25)

LINE(0,25,0,50)

LINE(50,25,50,50)

LINE(25,0,25,25)

LINE(0,0,50,0)

FOR X=0 TO GWIDTH STEP 50 LOOP

FOR Y=0 TO GHEIGHT STEP 50 LOOP

COPYREGION(0,0,50,50,X,Y)

REPEAT

REPEAT

UPDATE

Associated[GRABREGION](#)

COS

Purpose

Returns the cosine of the given angle.

Syntax

cosine=COS(*angle*)

Description

Returns the cosine of the argument *angle*. This is the ratio of the side of a right angled triangle, that is adjacent to the angle, to the hypotenuse (the longest side).

Example

```
CLS
PRINT "Draw ellipse in screen centre"
DEG
FOR Angle=0 TO 360 LOOP
    Xpos=100*COS(Angle)+GWIDTH / 2
    Ypos=50*SIN(Angle)+GHEIGHT / 2
    PLOT(Xpos, Ypos)
REPEAT
END
```

Associated

ACOS, ASIN, ATAN, SIN, TAN

LOOP

Purpose

Defines the start of a block of code to be repeated.

Syntax

LOOP

Description

Marks the start of a block of repeating code (called a loop). The number of times that the loop is executed depends on the command used before LOOP or at the end of the loop.

Example

REM See Associated commands below.

Associated

BREAK, CONTINUE, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

LOOP REPEAT

Purpose

Create an infinite loop.

Syntax

LOOP

```
{statements}
```

REPEAT

Description

Execute the *statements* again and again forever. The BREAK command can be used to terminate a loop. Pressing the Esc key will also interrupt the loop (and program).

Example

```
REM Loop until the space bar is pressed
PRINT "Press the space bar to continue"
LOOP
    IF INKEY = 32 THEN BREAK
REPEAT
END
```

Associated

[BREAK](#), [CONTINUE](#), [LOOP](#), [FOR REPEAT](#), [REPEAT UNTIL](#),
[UNTIL REPEAT](#), [WHILE REPEAT](#)



DATA

Purpose

Store constant data.

Syntax

```
DATA constant{constant}
```

Description

Stores numerical and string constants for later retrieval using the READ command.

Example

```
REM Load the name of the days of the
REM week into a string array
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday"
DATA "Sunday"
DIM DaysOfWeek$(7)
FOR DayNo=1 TO 7 LOOP
    READ DaysOfWeek$(DayNo)
REPEAT
FOR DayNo=1 TO 7 LOOP
    PRINT "Day of the week number ";DayNo;
    PRINT " is ";DaysOfWeek$(DayNo)
REPEAT
END
```

Associated

[READ](#), [RESTORE](#)



DATE\$

Purpose

Return the current date.

Syntax

todaydate\$ = DATE\$

Description

This returns a string with the current date in the format: YYYY-MM-DD. For example: 2015-03-24.

Example

```
PRINT "Today is ";FN FormatDate(DATE$)
END
```

```
DEF FN FormatDate()
  DayNo=VAL(RIGHT$(DATE$, 2))
  MonthNo=VAL(MID$(DATE$, 5, 2))
  Year$=LEFT$(DATE$,4)
  SWITCH (DayNo MOD 10)
    CASE 1
      DaySuffix$ = "st"
    ENDCASE
    CASE 2
      DaySuffix$ = "nd"
    ENDCASE
    CASE 3
      DaySuffix$ = "rd"
    ENDCASE
    DEFAULT
      DaySuffix$ = "th"
    ENDCASE
  ENDSWITCH
  FOR I=1 TO MonthNo LOOP
    READ MonthName$
  REPEAT
    Result$=STR$(DayNo)+DaySuffix$+" "
  =Result$+MonthName$+" "+Year$
  DATA "January","February","March","April"
  DATA "May","June","July","August"
  DATA "September","October","November"
  DATA "December"
```

Associated

[TIME\\$](#)



DEF FN

Purpose

Create a user defined function.

Syntax

```
DEF FN name({parameter},{parameter})
    {commands}
= value
```

Description

User defined functions are similar to user defined procedures except that they can return a value. This can be either a number or a character string.

Example

```
REM Function test - print squares
FOR I=1 TO 10 LOOP
    x=FN square(I)
PRINT I; " squared is "; x
REPEAT
END
DEF FN square(num)
    LOCAL result
    result=num*num
=result
```

Associated

FN, LOCAL



DEF PROC

Purpose

Create a user defined procedure.

Syntax

```
DEF PROC name({parameter},{parameter})
    {commands}
ENDPROC
```

Description

Allows you to create your own routines that can be called by their label. Once you have written a procedure to do a particular task you can copy it into other programs that require it. Procedures are usually defined after the END of the program.

Example

```
CLS
PROC Hexagon(200,200,100,Red)
UPDATE
END
DEF PROC Hexagon(x,y,l,c)
    PENUP
    MOVETO(x-l*COS(30),y+l/2)
    COLOUR = c
    PENDOWN
    FOR I=1 to 6 LOOP
        RIGHT(60)
        MOVE(I)
    REPEAT
ENDPROC
```

Associated

LOCAL, PROC, ENDPROC



DEFCHAR

Purpose

Define a new font character.

Syntax

DEFCHAR(*char*,*line1* ... *line10*)

Description

Create a user defined font character. The *char* parameter is the position of the character within the font (0-255) e.g. 65 is the capital letter A in ASCII. The character consists of 10 lines with 8 pixels in each line. These are set by the corresponding bits in each of the *line* parameters. So for example decimal 170 (binary 10101010) would set alternate pixels on the corresponding line of the character.

NOTE: The zero 'CHR\$(0)' should not be used.

Example

```
REM Define Chessboard Character
FONTSIZE(10)
DEFCHAR(2,0,85,170,85,170,85,170,85,170,0)
PRINT CHR$(2)
END
```

Associated

CHR\$



DEG

Purpose

Set angle units to degrees.

Syntax

DEG

Description

Switches the internal angle system to degree mode. There are 360 degrees in a full circle.

Example

```
REM Draw an ellipse in the screen centre
CLS
DEG
FOR Angle = 0 TO 360 LOOP
    Xpos=100*COS(Angle)+GWIDTH / 2
    Ypos=50*SIN(Angle)+GHEIGHT / 2
    PLOT(Xpos, Ypos)
REPEAT
END
```

Associated

CLOCK, RAD



DIGITALREAD

Purpose

Read the state of a digital pin on the Raspberry Pi.

Syntax

pinvalue=DIGITALREAD(*pinno*)

Description

Reads the state of a digital pin on the Raspberry Pi. You may need to set the pin mode beforehand to make sure it's configured as an input device. It will return TRUE or FALSE to indicate an input being high or low respectively.

Example

```
REM Set pin 12 to input
PINMODE(12,0)
REM Wait for button to be pushed
UNTIL DIGITALREAD(12) LOOP
REPEAT
PRINT "Button Pushed"
END
```

Associated

[DIGITALWRITE](#), [PINMODE](#), [PWMWRITE](#)



DIGITALWRITE

Purpose

Set the state of a digital pin on the Raspberry Pi.

Syntax

DIGITALWRITE(*pinno*,*pinvalue*)

Description

This procedure sets a digital pin to the supplied value - 0 for off or 1 for on. As with DigitalRead, you may need to set the pin mode (to output) beforehand.

Example

```
REM Flash LED attached to pin2 of GPIO
REM Set pin 2 to output mode
PINMODE(2,1)
LOOP
    REM Set output High (on)
    DIGITALWRITE(2,1)
    WAIT(1)
    REM Set output Low (off)
    DIGITALWRITE(2,0)
    WAIT(1)
REPEAT
END
```

Associated

[DIGITALREAD](#), [PINMODE](#), [PWMWRITE](#)



DIM

Purpose

Dimension an array of variables.

Syntax

DIM *variable*(dimension{,dimension})

Description

Creates an indexed variable with one or more dimensions. The variable can be either a numeric or character string type (they cannot hold mixed values). The index is a number from 0 to the size of the dimension. Associative arrays (sometimes called maps) are another way to refer to the individual elements of an array. In the example below we use a number, however strings are also allowed. They can be multi-dimensional and you can freely mix numbers and strings for the array indices.

Example

```
REM Initialise the squares of a chess
REM board to black or white
DIM ChessBoard(8,8)
Count=0
FOR Row=1 TO 8 LOOP
  FOR Col=1 TO 8 LOOP
    Count=Count+1
    IF Count MOD 2=1 THEN
      ChessBoard(Row,Col) = Black
    ELSE
      ChessBoard(Row,Col) = White
    ENDIF
  REPEAT
REPEAT
PRINT ChessBoard(1,4)
END
```

DRCANALOGREAD

Purpose

Read an analog channel on a DRC device.

Syntax

voltage=DRCANALOGREAD(*handle*,*pin*)

Description

This function reads an analog channel on a DRC compatible device specified by *handle* and returns the result. The value returned will depend on the hardware you're connected to - for example the Arduino will return a number from 0 to 1023 representing an input voltage between 0 and 5 volts. Other devices may have different ranges.

Example

```
arduino=DRCOPEN("/dev/ttyUSB0")
REM Get voltage on pin 4
voltage=DRCANALOGREAD(arduino, 4)/1023*5
PRINT "Voltage= "; voltage
DRCCLOSE(arduino)
END
```

Associated

DRCCLOSE, DRCDIGITALREAD, DRCDIGITALWRITE,
DRCOPEN, DRCPINMODE, DRCPWMWRITE



DRCCLOSE

Purpose

Close a connection to a DRC compatible device.

Syntax

DRCCLOSE(*handle*)

Description

This closes a connection to a DRC device and frees up any resources used by it. It's not strictly necessary to do this when you end your program, but it is considered good practice.

Example

```
arduino = DRCOPEN("/dev/ttyUSB0")
REM Set pin 12 to input
DRCPINMODE(arduino, 12, 0)
LOOP
REPEAT UNTIL DRCDIGITALREAD(arduino, 12)
PRINT "Button Pushed"
DRCCLOSE(arduino)
END
```

Associated

DRCANALOGREAD, DRCDIGITALREAD, DRCDIGITALWRITE,
DRCOPEN, DRCPINMODE, DRCPWMWRITE



DRCDIGITALREAD

Purpose

Read the state of a digital pin on a remote DRC device.

Syntax

`state=DRCDIGITALREAD(handle,pin)`

Description

This function allows you to read the state of a digital pin on a DRC device specified by *handle*. You may need to set the pin mode beforehand to make sure it's configured as an input device. It will return TRUE or FALSE to indicate an input being high or low respectively.

Example

```
arduino = DRCOPEN("/dev/ttyUSB0")
REM Set pin 12 to input
DRCPINMODE(arduino, 12, 0)
LOOP
REPEAT UNTIL DRCDIGITALREAD(arduino, 12)
PRINT "Button Pushed"
DRCCLOSE(arduino)
END
```

Associated

[DRCANALOGREAD](#), [DRCCLOSE](#), [DRCDIGITALWRITE](#),
[DRCOPEN](#), [DRCPINMODE](#), [DRCPWMWRITE](#)

DRCDIGITALWRITE

Purpose

Set a digital pin on a remote DRC device to the supplied value.

Syntax

`DRCDIGITALWRITE(handle,pin,value)`

Description

This procedure sets a digital *pin* on a DRC device specified by *handle* to the supplied *value* - 0 for off or 1 for on. As with [DrcDigitalRead](#), you may need to set the pin mode beforehand.

Example

```
arduino=DRCOPEN("/dev/ttyUSB0")
REM Set pin 2 to output mode
DRCPINMODE(arduino, 2, 1)
REM Set Output High (on)
DRCDIGITALWRITE(arduino, 2, 1)
REM Pause for 1 second
WAIT(1)
REM Set output Low (off)
DRCDIGITALWRITE(arduino, 2, 0)
DRCCLOSE(arduino)
END
```

Associated

[DRCANALOGREAD](#), [DRCCLOSE](#), [DRCDIGITALREAD](#),
[DRCOPEN](#), [DRCPINMODE](#), [DRCPWMWRITE](#)

DRCOPEN

Purpose

Open a connection to a DRC compatible device.

Syntax

handle=DRCOPEN(*drcdevice*)

Description

This opens a connection to a DRC compatible device and makes it available for our use. It takes the name of the device as an argument and returns a number (the handle) of the device. We can use this handle to reference the device and allow us to open several devices at once. Some implementations may have IO devices with fixed names.

Example

```
arduino = DRCOPEN("/dev/ttyUSB0")
REM Set pin 12 to input
DRCPINMODE(arduino, 12, 0)
LOOP
REPEAT UNTIL DRCDIGITALREAD(arduino, 12)
PRINT "Button Pushed"
DRCCLOSE(arduino)
END
```

Associated

[DRCANALOGREAD](#), [DRCCLOSE](#), [DRCDIGITALREAD](#),
[DRCDIGITALWRITE](#), [DRCPINMODE](#), [DRCPWMWRITE](#)



DRCPINMODE

Purpose

Configure the mode of a pin on a remote DRC device.

Syntax

DRCPINMODE(*handle*,*pin*,*mode*)

Description

This configures the mode of a pin on the DRC device specified by *handle*. It takes an argument which specifies the *mode* of the specified *pin* - input, output or PWM output. Other modes may be available, depending on the device and its capabilities. Note that not all devices support all functions. The modes are:

0 pinINPUT

1 pinOUTPUT

2 pinPWM

Example

```
arduino = DRCOPEN("/dev/ttyUSB0")
REM Set pin 12 to input
DRCPINMODE(arduino, 12, 0)
LOOP
REPEAT UNTIL DRCDIGITALREAD(arduino, 12)
PRINT "Button Pushed"
DRCCLOSE(arduino)
END
```

Associated

[DRCANALOGREAD](#), [DRCCLOSE](#), [DRCDIGITALREAD](#),
[DRCDIGITALWRITE](#), [DRCOPEN](#), [DRCPWMWRITE](#)



DRCPWMWRITE

Purpose

Output a PWM waveform on the selected pin of a DRC device.

Syntax

DRCPWMWRITE(*handle*,*pin*,*value*)

Description

This procedure outputs a PWM waveform on the specified *pin* of a DRC compatible device specified by *handle*. The pin must be configured for PWM mode beforehand, and depending on the device you are using, then not all pins on a device may support PWM mode. The *value* set should be between 0 and 255.

Example

```
arduino=DRCOPEN("/dev/ttyUSB0")
REM Set pin 11 to PWM output mode
DRCPINMODE(arduino, 11, 2)
DRCPWMWRITE(arduino, 11, 200)
DRCCLOSE(arduino)
END
```

Associated

DRCANALOGREAD, DRCCLOSE, DRCDIGITALREAD,
DRCDIGITALWRITE, DRCOPEN, DRCPINMODE



ELLIPSE

Purpose

Draw an ellipse on the screen.

Syntax

ELLIPSE(*xpos*,*ypos*,*xradius*,*yradius*,*fill*)

Description

Draws an ellipse centred at position (*xpos*,*ypos*) with the specified *xradius* and *yradius* in the current foreground COLOUR. The final parameter *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Example

```
CLS
REM Draw a filled red ellipse at
REM location 200,200
COLOUR=red
ELLIPSE(200,200,100,50,TRUE)
UPDATE
END
```

Associated

CIRCLE, RECT, TRIANGLE



ELSE

Purpose

Execute statement(s) when a tested condition is False.

Syntax

IF *condition* THEN

{*statements*}

ELSE

{*statements*}

ENDIF

Example

Number = 13

IF Number MOD 2 = 0 THEN

PRINT "Number is Even"

ELSE

PRINT "Number is Odd"

ENDIF

END

Associated

ENDIF, IF THEN

END

Purpose

End program execution.

Syntax

END

Description

Program execution is ended. Programs must terminate with the END or STOP commands or an error will occur.

Example

PRINT "Hello World"

END

ENDIF

Purpose

Terminate a multiline conditional statement.

Syntax

```
IF condition THEN  
{statements}  
ENDIF
```

Description

We can extend the IF statement over multiple lines, if required. The way you do this is by making sure there is nothing after the THEN statement and ending it all with the ENDIF statement.

Example

```
DayOfWeek = 5  
IF DayOfWeek < 6 THEN  
    PRINT "It is a Weekday"  
    PRINT "Go to Work!"  
ENDIF  
END
```

Associated

ELSE, IF THEN

ENDPROC

Purpose

Defines the end of a PROCedure

Syntax

```
ENDPROC
```

Description

End a PROCedure and return to the next command after the procedure was called.

Example

```
CLS  
PROC hello  
END  
DEF PROC hello  
PRINTAT (10,10); "Hello"  
ENDPROC
```

Associated

PROC, DEF PROC

ENVELOPE

Purpose

Emulate the BBC BASIC sound envelope command.

Syntax

ENVELOPE(*N,T,PI1,P12,PI3,PN1,PN2,PN3,AA,AD,AS,AR,ALA,ALD*)

Description

NOTE: This is an experimental function. It might not perform entirely as expected. It is also prone to crashing if incorrect values are used. Use with caution!

N 1 to 8 Envelope number.

T 0 to 127 Length of each step in hundredths of a second.

Add 128 to cancel auto repeat of the pitch envelope.

PI1 –128 to 127 Pitch change per step in section 3

PI2 –128 to 127 Change of pitch per step in section 2

PI3 –128 to 127 Change of pitch per step in section 1

PN1 0 to 255 Number of steps in section 1

PN2 0 to 255 Number of steps in section 2

PN3 0 to 255 Number of steps in section 3

AA –127 to 127 Change of attack amplitude per step

AD –127 to 127 Change of decay amplitude per step

AS –127 to 0 Change of sustain amplitude per step

AR –127 to 0 Change of release amplitude per step

ALA 0 to 126 Level at end of the attack phase

ALD 0 to 126 Level at end of the decay phase

Example

```
ENVELOPE(1,2,-  
2,10,1,80,40,40,127,0,0,0,126,126)  
SOUND(1, 1, 53, 64)  
END
```

Associated

[SOUND](#)

EOF

Purpose

Return true if the end of an input file has been reached.

Syntax

endoffile=EOF(*handle*)

Description

The EOF function returns a TRUE or FALSE indication of the state of the file pointer when reading the file. It is an error to try to read past the end of the file, so if you are reading a file with unknown data in it, then you must check at well defined intervals (e.g. Before each INPUT#).

Example

```
handle=OPEN("eoftest.txt")
FOR r = 0 TO 10 LOOP
    PRINT# handle, "Record "; r
REPEAT
CLOSE (handle)
handle = OPEN("eoftest.txt")
WHILE NOT EOF (handle) LOOP
    INPUT# handle, record$
    PRINT record$
REPEAT
CLOSE (handle)
END
```

Associated

CLOSE, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK



EXP

Purpose

Return the exponential value of the specified number.

Syntax

exponential=EXP(*number*)

Description

Returns the exponential value of the specified *number*. This is e to the power of *number* where e is the exponential constant (approximately 2.718281828). The exponential function arises whenever a quantity grows or decays at a rate proportional to its current value. This is the opposite of the LOG function i.e. EXP(LOG(X)) = X

Example

```
REM prints 2.718281828
PRINT EXP(1)
REM prints 22026.46579
PRINT EXP(10)
REM prints 10
PRINT LOG(EXP(10))
END
```

Associated

LOG



FADEOFF

Purpose

Fade the display from light to dark

Syntax

FADEOFF

Description

FADEOFF initiates a fade from light to dark. The entire screen display is affected.

Example

```
PAPER=0
INK=1
CLS
PRINT "LOADING..."
FADEOFF
WHILE FADING = true LOOP
    UPDATE
REPEAT
FADEON
WHILE FADING = true LOOP
    UPDATE
REPEAT
END
```

Associated

[FADEON](#), [FADING](#)



FADEON

Purpose

Fade the display from dark to light

Syntax

FADEON

Description

FADEON initiates a fade from dark to light. The entire screen display is affected.

Example

```
PAPER=0
INK=1
CLS
PRINT "LOADING..."
FADEOFF
WHILE FADING = true LOOP
    UPDATE
REPEAT
FADEON
WHILE FADING = true LOOP
    UPDATE
REPEAT
END
```

Associated

[FADEOFF](#), [FADING](#)



FADING

Purpose

Check if the display is fading

Syntax

FADING

Description

Returns either TRUE (if fade in progress) or FALSE (no fade active)

Example

```
PAPER=0
INK=1
CLS
PRINT "LOADING..."
FADEOFF
WHILE FADING = true LOOP
    UPDATE
REPEAT
FADEON
WHILE FADING = true LOOP
    UPDATE
REPEAT
END
```

Associated

FADEON, FADING



FALSE

Purpose

Represent the logical "false" value.

Syntax

FALSE

Description

Represents a Boolean value that fails a conditional test. It is equivalent to a numeric value of 0.

Example

```
condition = FALSE
IF condition = FALSE THEN
    PRINT "Condition is FALSE"
ENDIF
IF NOT condition THEN
    PRINT "Condition is FALSE"
ENDIF
PRINT "Condition= ";condition
END
```

Associated

TRUE



FFWD

Purpose

Move the file pointer to the end of a file.

Syntax

FFWD(*handle*)

Description

Move the file pointer back to the end of the file specified by *handle*. If you want to append data to the end of an existing file, then you need to call FFWD before writing the data.

Example

```
handle=OPEN("ffwdtest.txt")
PRINT# handle, "First Line"
CLOSE (handle)
handle = OPEN ("ffwdtest.txt")
FFWD (handle)
PRINT# handle, "Appended line"
CLOSE (handle)
handle = OPEN("ffwdtest.txt")
WHILE NOT EOF (handle) LOOP
    INPUT# handle, record$
    PRINT record$
REPEAT
CLOSE (handle)
END
```

Associated

CLOSE, EOF, INPUT#, OPEN, PRINT#, REWIND, SEEK



FN

Purpose

Call a user defined function.

Syntax

result=FN *name*({*argument*},{*argument*})

Description

Calls the specified user defined function called *name* with the specified *arguments*. The returned *result* can then be used by the program. Once the function has been executed control returns to the command following.

Example

```
PRINT FN SphereVolume(10)
END
REM Function calculate volume of a sphere
REM with radius r
DEF FN SphereVolume(r)
    = (4/3)*PI*r*r*r
```

Associated

DEF FN



FONTSIZE

Purpose

Scale the text font.

Syntax

FONTSIZE(*scale*)

Description

Change the size of the text font.

Example

```
FOR S=1 TO 10 LOOP
    FONTSIZE(S)
    PRINT "Hello World"
REPEAT
END
```

Associated

PRINTAT, LOADFONT

FOR LOOP

Purpose

Loop a specified number of times using a counter.

Syntax

```
FOR count=start TO end [STEP step] LOOP
    statements
REPEAT
```

Description

The *count* variable is initially set to *start* and changes by *step* each time around the loop until *count* is greater than or equal to *end*. The optional *step*, which defaults to 1 may be less than zero to count backwards. The end of the loop is indicated using the REPEAT.

Example

```
REM year into a string array
DATA "January", "February", "March"
DATA "April", "May", "June"
DATA "July", "August", "September"
DATA "October", "November", "December"
DIM Months$(12)
FOR Month = 1 TO 12 LOOP
    READ Months$(Month)
REPEAT
PRINT "The seventh month is ";Months$(7)
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, REPEAT UNTIL,
UNTIL REPEAT, WHILE REPEAT



FREEIMAGE

Purpose

Release an image from memory

Syntax

FREEIMAGE(*handle*)

Description

Frees up the memory space taken up by a stored image

Example

```
handle = LOADIMAGE( "sprite1.bmp" )
PLOTIMAGE( handle, 0, 0 )
UPDATE
FREEIMAGE( handle )
handle = LOADIMAGE( "sprite2.bmp" )
PLOTIMAGE( handle, 100, 100 )
UPDATE
END
```

Associated

[LOADIMAGE](#), [PLOTIMAGE](#)

FULLSCREEN

Purpose

Sets the display to full screen mode.

Syntax

FULLSCREEN={TRUE/FALSE}

Description

Switches between full screen or windowed mode. Note this does not set the resolution to the screen display mode so unless you set the mode manually you will get a border.

Example

```
SETMODE(800,600)
FULLSCREEN=0
COLOUR=RED
RECT(0,0,GWIDTH, GHEIGHT,0)
UPDATE
WAIT(2)
FULLSCREEN=1
WAIT(2)
END
```

Associated

[SETMODE](#)

GET

Purpose

Get a single character code from the keyboard.

Syntax

```
asciicode=GET
```

Description

This pauses program execution and waits for you to type a single character on the keyboard, then returns the value of the key pressed as a numeric variable (ASCII).

Example

```
PRINT "Press a key"  
key = GET  
PRINT "ASCII Value of key = "; key  
END
```

Associated

GET\$, INKEY



GET\$

Purpose

Get a single character from the keyboard.

Syntax

```
key$ = GET$
```

Description

This pauses program execution and waits for you to type a single character on the keyboard, then returns the key as a string variable.

Example

```
PRINT "Press a key"  
key$ = GET$  
PRINT "You Pressed Key: "; key$  
END
```

Associated

GET, INKEY



GETIMAGEH

Purpose

Get the pixel height of a loaded image.

Syntax

GETIMAGEH(*handle*)

Description

Gets the height in pixels of a loaded image

Example

```
REM Centre image on screen
logo=LOADIMAGE("/usr/share/fuze/logo.bmp")
imageW=GETIMAGEW(logo)
imageH=GETIMAGEH(logo)
X=(GWIDTH-imageW)/2
Y=(GHEIGHT-imageH)/2
PLOTIMAGE(logo,X,Y)
UPDATE
END
```

Associated

[GETIMAGEW](#), [LOADIMAGE](#), [PLOTIMAGE](#)



GETIMAGEW

Purpose

Get the pixel width of a loaded image.

Syntax

GETIMAGEW(*handle*)

Description

Gets the width in pixels of an image previously loaded using [LOADIMAGE](#) (using the *handle* returned by [LOADIMAGE](#)).

Example

```
REM Centre image on screen
logo=LOADIMAGE("/usr/share/fuze/logo.bmp")
imageW=GETIMAGEW(logo)
imageH=GETIMAGEH(logo)
X=(GWIDTH-imageW)/2
Y=(GHEIGHT-imageH)/2
PLOTIMAGE(logo,X,Y)
UPDATE
END
```

Associated

[GETIMAGEH](#), [LOADIMAGE](#), [PLOTIMAGE](#)



GETMOUSE

Purpose

Read values from an attached mouse

Syntax

GETMOUSE(*xpos,ypos,buttons*)

Description

This reads values for the current state of the mouse. *xpos* is the horizontal mouse position, *ypos* is the vertical position and *buttons* is the state of the mouse buttons. You can test whether the left button has been pressed by using the logical & operator to see if bit 0 of the *buttons* value is set: *buttons* & 1 will be TRUE. Likewise if the right button is pressed then bit 3 will be set and *buttons* & 4 will be TRUE.

Example

```
CLS
MOUSEON
LOOP
  GETMOUSE(x,y,b)
  LINETO(x,y)
  UPDATE
  REM LOOP colour if left button pressed
  IF b & 1 THEN
    COLOUR = COLOUR MOD 16 + 1
  ENDIF
  REM Exit if right button pressed
REPEAT UNTIL b & 4
MOUSEOFF
END
```

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, SETMOUSE



GETPIXEL

Purpose

Return the colour of the specified pixel.

Syntax

colour = GETPIXEL(*xpos, ypos*)

Description

This returns the internal colour code (0-15) of the pixel at the specified point (*xpos,ypos*). This is for the "named" colours - e.g. Red, Green etc. It returns -1 if the pixel colour is not a standard colour - in which case, you need to use GETPIXELRGB.

Example

```
CLS
COLOUR = RED
Xpos = GWIDTH / 2
Ypos = GHEIGHT / 2
PLOT(Xpos, Ypos)
PRINT GETPIXEL(Xpos, Ypos)
END
```

Associated

GETPIXELRGB



GETPIXELRGB

Purpose

Return the RGB colour of the specified pixel.

Syntax

RGBcolour = GETPIXELRGB(*xpos*, *ypos*)

Description

This returns the RGB colour of the pixel at the specified point (*xpos*,*ypos*). This will return a 24-bit value.

Example

```
CLS
RGBCOLOUR(49, 101, 206)
Xpos = GWIDTH / 2
Ypos = GHEIGHT / 2
CIRCLE(Xpos, Ypos, 50, TRUE)
pixel = GETPIXELRGB(Xpos, Ypos)
PRINT "RGB = ";      pixel
PRINT "Red = ";      (pixel >> 16) & 0xFF
PRINT "Green = ";    (pixel >> 8) & 0xFF
PRINT "Blue = ";     (pixel >> 0) & 0xFF
END
```

Associated

GETPIXEL

GETSPRITEANGLE

Purpose

Returns a sprite's current angle

Syntax

GETSPRITEANGLE(*spriteIndex*)

Description

Returns the angle of the sprite with the specified *spriteIndex*.

Example

```
sprite = newSprite (1)
loadSprite ("logo.png", sprite, 0)
angle = 0
plotSprite (sprite, gWidth / 2, gHeight / 2, 0)
LOOP
cls2
    setSpriteAngle (sprite, angle)
    angle = angle + 1
    if angle>359 THEN angle=0
    printat(0,0);getspriteangle(sprite)
    UPDATE
REPEAT
```

Associated

GETSPRITEW, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE, SPRITECOLLIDEP

GETSPRITEH

Purpose

Get the pixel height of a sprite.

Syntax

GETSPRITEH(*spriteIndex*)

Description

Returns the height in pixels of the sprite with the specified *spriteIndex*.

Example

```
REM Centre sprite on the screen
index=NEWSPRITE(1)
fuzelogo$="logo.bmp"
LOADSPRITE(fuzelogo$,index,0)
spriteW=GETSPRITEW(index)
spriteH=GETSPRITEH(index)
X=(GWIDTH-spriteW)/2
Y=(GHEIGHT-spriteH)/2
PLOTSPRITE(index,X,Y,0)
UPDATE
END
```

Associated

GETSPRITEW, HIDESPRITE, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE,
SPRITECOLLIDEPP



GETSPRITEW

Purpose

Get the pixel width of a sprite.

Syntax

GETSPRITEW(*spriteIndex*)

Description

Returns the width in pixels of the sprite with the specified *spriteIndex*.

Example

```
REM Centre sprite on the screen
index=NEWSPRITE(1)
fuzelogo$="logo.bmp"
LOADSPRITE(fuzelogo$,index,0)
spriteW=GETSPRITEW(index)
spriteH=GETSPRITEH(index)
X=(GWIDTH-spriteW)/2
Y=(GHEIGHT-spriteH)/2
PLOTSPRITE(index,X,Y,0)
UPDATE
END
```

Associated

GETSPRITEH, HIDESPRITE, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE,
SPRITECOLLIDEPP



GETSPRITEX

Purpose

Get the X position of a sprite.

Syntax

GETSPRITEX(*spriteIndex*)

Description

Returns the X position in pixels of the sprite with the specified *spriteIndex*.

Example

```
updateMode = 0
sprite = newSprite (1)
loadSprite ("logo.png", sprite, 0)
LOOP
    CLS2
    plotSprite (sprite, gWidth / 2, gHeight / 2, 0)
    PRINT getSpriteX (sprite)
    PRINT getSpriteY (sprite)
    UPDATE
REPEAT
```

Associated

GETSPRITEW, HIDESPRITE, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE,
SPRITECOLLIDEPP



GETSPRITEY

Purpose

Get the Y position of a sprite.

Syntax

GETSPRITEY(*spriteIndex*)

Description

Returns the Y position in pixels of the sprite with the specified *spriteIndex*.

Example

```
updateMode = 0
sprite = newSprite (1)
loadSprite ("logo.png", sprite, 0)
LOOP
    CLS2
    plotSprite (sprite, gWidth / 2, gHeight / 2, 0)
    PRINT getSpriteX (sprite)
    PRINT getSpriteY (sprite)
    UPDATE
REPEAT
```

Associated

GETSPRITEH, HIDESPRITE, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE,
SPRITECOLLIDEPP



GHEIGHT

Purpose

Find the current height of the display.

Syntax

height=GHEIGHT

Description

This can be read to find the current height of the display in either high resolution or low resolution pixels.

Example

```
REM Draw a circle in the centre of the
screen
CLS
COLOUR = blue
CIRCLE(GWIDTH/2,GHEIGHT/2,50,TRUE)
UPDATE
END
```

Associated

GWIDTH, ORIGIN, SETMODE

GRABREGION

Purpose

Grab a region of the screen to a temporary buffer

Syntax

handle = GRABREGION(*x*, *y*, *width*, *height*)

Description

Grab a region of the screen with *x* and *y* as the location and with *width* and *height* in pixels. The region can be recalled by its *handle* and pasted using PLOTIMAGE.

Example

```
FOR n = 0 TO 15 LOOP
RECT(0,n*GHEIGHT/16, GWIDTH, GHEIGHT/16,1)
REPEAT
handle=GRABREGION(0,0,200,200)
CLS
PLOTIMAGE(handle, GWIDTH/2, GHEIGHT/2)
UPDATE
END
```

Associated

COPYREGION, FREEIMAGE, LOADIMAGE, PLOTIMAGE,
SAVEREGION, SAVESCREEN, SCROLLDOWN, SCROLLLEFT,
SCROLLRIGHT, SCROLLUP

GWIDTH

Purpose

Returns the current width of the display.

Syntax

width=GWIDTH

Description

This can be read to find current width of the display in either high resolution or low resolution pixels.

Example

```
REM Draw a circle in the centre of the
screen
CLS
COLOUR = blue
CIRCLE(GWIDTH/2,GHEIGHT/2,50,TRUE)
UPDATE
END
```

Associated

GHEIGHT, ORIGIN, SETMODE

HIDESPRITE

Purpose

Remove a sprite from the screen.

Syntax

HIDESPRITE(*spriteindex*)

Description

This removes the sprite at the specified *spriteindex* from the screen. You do not have to erase a sprite from the screen when you move it, just call PLOTSPRITE with the new coordinates.

Example

```
CLS
fuzelogo$="logo.bmp"
s1=NEWSPRITE (1)
s2=NEWSPRITE (1)
LOADSPRITE (fuzelogo$,s1,0)
LOADSPRITE (fuzelogo$,s2,0)
PLOTSPRITE (s1,100,100,0)
PLOTSPRITE (s2,200,200,0)
UPDATE
WAIT(2)
REM Remove a sprite from the screen
HIDESPRITE (s2)
UPDATE
END
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT



HLINE

Purpose

Draw a horizontal line.

Syntax

HLINE(*xpos1*, *xpos2*, *ypos*)

Description

Draws a horizontal line on row *y*, from column *xpos1* to column *xpos2*.

Example

```
CLS
COLOUR=red
FOR ypos=0 TO GHEIGHT STEP 100 LOOP
    HLINE(0,GWIDTH,ypos)
REPEAT
UPDATE
END
```

Associated

LINE, LINETO, VLINE

HTAB

Purpose

Set/Read the current text cursor horizontal position.

Syntax

HTAB=*value*

value=HTAB

Description

Set/Read the current text cursor horizontal position.

Example

```
CLS
FOR xpos = 0 TO TWIDTH STEP 2 LOOP
    HTAB=xpos
    PRINT HTAB
REPEAT
END
```

Associated

HVTAB, VTAB, PRINTAT

HVTAB

Purpose

Move the current text cursor to the specified position.

Syntax

HVTAB(*xpos*,*ypos*)

Description

The cursor is moved to the supplied column *xpos* and line *ypos*. Note that (0,0) is top-left of the text screen.

Example

```
CLS
HVTAB(TWIDTH/2-3,THEIGHT/2)
PRINT "CENTRE"
HVTAB(0,0)
END
```

Associated

HTAB, VTAB, PRINTAT

IF THEN

Purpose

Execute a statement conditionally.

Syntax

IF *condition* THEN {*statement*}

Description

The statement is executed when the condition evaluates to TRUE (not 0). Unlike some implementations of BASIC the THEN is required.

Example

```
PRINT "Press Space Bar to Continue"
LOOP
    IF INKEY = 32 THEN BREAK
REPEAT
PRINT "Space Bar Pressed"
END
```

Associated

ELSE, ENDIF, SWITCH

INK

Purpose

Set/Read the current text foreground colour.

Syntax

foregroundcolour=INK

INK=*foregroundcolour*

Description

Set/Read the current text foreground (ink) colour.

Example

PAPER=black

CLS

string\$="This is multicoloured text"

FOR I=1 TO LEN(string\$) LOOP

 INK=I MOD 15 + 1

 PRINT MID\$(string\$,I-1,1);

REPEAT

INK=white

PRINT

END

Associated

PAPER

INKEY

Purpose

Get a single character code from the keyboard without pausing.

Syntax

*ascii*code=INKEY

Description

This is similar to GET except that program execution is not paused; If no key is pressed, then -1 is returned. The following constants are predefined to test for special keys: KeyUp, KeyDown, KeyLeft, KeyRight, KeyIns, KeyHome, KeyDel, KeyEnd, KeyPgUp, KeyPgDn, KeyF1, KeyF2, KeyF3, KeyF4, KeyF5, KeyF6, KeyF7, KeyF8, KeyF9, KeyF10, KeyF11, KeyF12.

Example

REM Show the ASCII code for the last key pressed

LastKey = -1

REM Press Esc to Quit

LOOP

 Key=INKEY

 IF Key<>-1 AND Key<>LastKey THEN

 PRINT "Key Pressed: "; Key

 LastKey = Key

 ENDIF

REPEAT

Associated

GET, INPUT, SCANKEYBOARD



INPUT

Purpose

Read data from the keyboard into a variable.

Syntax

INPUT [*prompt*,\$] *variable*

Description

When FUZE BASIC encounters the INPUT statement, program execution stops, a question mark(?) is printed and it waits for you to type something. It then assigns what you typed to the variable. If you typed in a string when it was expecting a number, then it will assign zero to the number. To stop it printing the question mark, you can optionally give it a string to print.

Example

```
INPUT "What is your Name? ", Name$  
PRINT "Hello " + Name$  
END
```

Associated

INKEY



INPUT#

Purpose

Read data from a file.

Syntax

INPUT# *handle*,*variable*

Description

This works similarly to the regular INPUT instruction, but reads from the file identified by the supplied *handle* rather than from the keyboard. Note that unlike the regular keyboard INPUT instruction, INPUT# can only read one variable at a time.

Example

```
handle=OPEN("testfile.txt")  
PRINT# handle,"Hello World"  
REWIND(handle)  
INPUT# handle,record$  
PRINT record$  
CLOSE (handle)  
END
```

Associated

CLOSE, EOF, FFW, OPEN, PRINT#, REWIND, SEEK



INT

Purpose

Return the integer part of a number.

Syntax

integerpart=INT(*number*)

Description

Returns the integer part of the specified *number*.

Example

```
PRINT "The integer part of PI is ";  
PRINT INT(PI)  
END
```

LEFT

Purpose

Turns the turtle to the left (counter clockwise) by the given angle.

Syntax

LEFT(*angle*)

Description

Turns the virtual graphics turtle to the left (counter clockwise) by the given *angle* in the current angle units.

Example

```
REM Draw a box using turtle graphics  
CLS  
COLOUR=RED  
MOVE(50)  
DEG  
LEFT(90)  
MOVE(50)  
PENDOWN  
FOR I = 1 TO 4 LOOP  
    LEFT(90)  
    MOVE(100)  
REPEAT  
UPDATE  
END
```

Associated

MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE



LEFT\$

Purpose

Return the specified leftmost number of a characters from a string.

Syntax

```
substring$=LEFT$(string$,number)
```

Description

Returns a substring of *string\$* with *number* characters from the left (start) of the string. If number is greater than or equal to the length of *string\$* then the whole string is returned.

Example

```
string$="The quick brown fox"  
FOR I=1 TO 20 LOOP  
    PRINT LEFT$(string$, I)  
REPEAT  
END
```

Associated

MID\$, RIGHT\$

LEN

Purpose

Return the length of the specified character string.

Syntax

```
length=LEN(string$)
```

Description

Returns the number of characters in the specified *string\$*.

Example

```
String$="The Quick Brown Fox"  
Chars=LEN(String$)  
PRINT "String Length is: ";Chars  
FOR I=0 TO Chars - 1 LOOP  
    Char$=MID$(String$,I,1)  
    PRINT "Character No. ";I," is "+Char$  
REPEAT  
END
```

LINE

Purpose

Draw a line between two points

Syntax

LINE(*xpos1*,*ypos1*,*xpos2*,*ypos2*)

Description

Draw a line between point (*xpos1*,*ypos1*) and point (*xpos2*,*ypos2*) in the current COLOUR.

Example

```
CLS
COLOUR = lime
GH=GHEIGHT
GW=GWIDTH
LINE(10,10,10,GH-10)
LINE(10,GH-10,GW-10,GH-10)
LINE(GW-10,GH-10,GW-10,10)
LINE(GW-10,10,10,10)
UPDATE
END
```

Associated

HLINE, LINETO, VLINE

LINETO

Purpose

Draw a line from the last point plotted.

Syntax

LINETO(*xpos1*,*ypos1*)

Description

Draws a line from the last point plotted (by the PLOT or LINE procedures) to point (*xpos1*,*ypos1*).

Example

```
CLS
COLOUR = yellow
ORIGIN(10,10)
GH=GHEIGHT
GW=GWIDTH
LINETO(0,GH-20)
LINETO(GW-20,GH-20)
LINETO(GW-20, 0)
LINETO(0,0)
UPDATE
END
```

Associated

HLINE, LINE, VLINE

LOADIMAGE

Purpose

Load an image file into memory.

Syntax

handle=LOADIMAGE(*filename*)

Description

Load an image from a file with the specified *filename*. The returned *handle* can then be used to plot it on the screen with PLOTIMAGE.

Example

```
COLOUR=RED
RECT(0,0,50,50,TRUE)
COLOUR=WHITE
LINE(0,50,50,50)
LINE(0,25,50,25)
LINE(0,25,0,50)
LINE(50,25,50,50)
LINE(25,0,25,25)
LINE(0,0,50,0)
SAVEREGION("bricks.bmp",0,0,50,50)
handle=LOADIMAGE("bricks.bmp")
FOR X=0 TO GWIDTH STEP 50 LOOP
  FOR Y=0 TO GHEIGHT STEP 50 LOOP
    PLOTIMAGE(handle,X,Y)
  REPEAT
REPEAT
UPDATE
END
```

Associated

GETIMAGEH, GETIMAGEW, PLOTIMAGE, FREEIMAGE



LOADMUSIC

Purpose

Load a music file into memory ready to be played.

Syntax

handle=LOADMUSIC(*filename*)

Description

Loads an uncompressed music file in wav format (file extension .wav) from the file *filename* into memory and returns a *handle* which can then be used to play the music using the PLAYMUSIC function.

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
END
```

Associated

PAUSEMUSIC, RESUMEMUSIC, SETMUSICVOL, STOPMUSIC



LOADSAMPLE

Purpose

Load a sound sample into memory ready to be played.

Syntax

handle=LOADSAMPLE(*filename*)

Description

The LOADSAMPLE function loads a sound sample from the uncompressed WAV format file called *filename* and returns a *handle* to it so that it can be played using the PLAYSAMPLE function. You can load up to 32 sound samples into memory at the same time.

Example

```
channel=0
volume=70
SETCHANVOL(channel,volume)
intro=LOADSAMPLE("pacman_intro.wav")
PLAYSAMPLE(intro,channel,0)
WAIT(4.5)
END
```

Associated

PAUSECHAN, PLAYSAMPLE, RESUMECHAN, SETCHANVOL, STOPCHAN

LOADSPRITE

Purpose

Load a sprite from a file into memory.

Syntax

LOADSPRITE(*filename*,\$*index*,*subindex*)

Description

This loads a sprite from the supplied *filename*\$ into memory and associates it with the given sprite *index* and *subindex*. The *index* is the handle returned by a call to NewSprite and the *subindex* is the version of the sprite to allow for animation. The first *subindex* is 0.

Example

```
CLS
REM Create a new sprite with 1 version
SpriteIndex=NEWSPRITE(1)
REM Load a sprite from a file
fuzelogo$="logo.bmp"
LOADSPRITE(fuzelogo$,SpriteIndex,0)
REM Draw the sprite on the screen
PLOTSPRITE(SpriteIndex,200,200,0)
UPDATE
END
```

Associated

GETSPRITEH, GETSPRITEW, HIDESPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE, SPRITECOLLIDEPP



LOCAL

Purpose

Define variables to be local to a user defined procedure or function.

Syntax

LOCAL *variable*

Description

Allows a variable name to be reused in a procedure or function without affecting its value in the calling program.

Example

```
X=10
PRINT "Global X="+STR$(X)
Proc Test()
PRINT "Global X="+STR$(X)
END
DEF PROC Test()
    PRINT "Global X="+STR$(X)
    LOCAL X
    X=5
    PRINT "Local X="+STR$(X)
ENDPROC
```

Associated

[DEF FN](#), [DEF PROC](#)



LOG

Purpose

Return the natural logarithm of the specified number.

Syntax

naturallogarithm = LOG(*number*)

Description

Returns the natural logarithm of the specified *number*. This is the opposite of the EXP function i.e. LOG(EXP(X)) = X. Logarithms are used in science to solve exponential radioactive decay problems and in finance to solve problems involving compound interest.

Example

```
X = EXP(10)
PRINT X // Will print 22026.46579
PRINT LOG(X) // Will print 10
END
```

Associated

[EXP](#)



MAX

Purpose

Returns the larger of two numbers.

Syntax

maxvalue=MAX(*number1*,*number2*)

Description

Returns the larger (highest value) of *number1* or *number2*.

Example

```
REM Prints the value of number2
number1=12.26
number2=27.45
PRINT MAX(number1, number2)
END
```

Associated

[MIN](#)

MICROTIME

Purpose

Find out how long the program has been running in 1,000,000s of a second.

Syntax

time=MICROTIME

Description

This returns a number which represents the time that your program has been running in millionths of seconds.

Example

```
REM Simple reaction timer
WAIT(2)
REM Make sure no key pressed
WHILE INKEY<>-1 LOOP
REPEAT
stime=MICROTIME
PRINT "Go!"
WHILE INKEY=-1 LOOP
REPEAT
etime = MICROTIME
PRINT "Your reaction time is ";
PRINT etime-stime; " milliseconds"
END
```

MID\$

Purpose

Return characters from the middle of a string.

Syntax

MID\$(*string\$*, start, length)

Description

Returns the middle *length* characters of *string\$* starting from position *start*. The first character of the string is position number 0.

Example

```
REM Prints Quick  
string$="The Quick Brown Fox"  
PRINT MID$(string$,4,5)  
END
```

Associated

LEFT\$, RIGHT\$

MIN

Purpose

Returns the smaller of two numbers.

Syntax

minvalue=MIN(*number1*,*number2*)

Description

Returns the smaller (lowest value) of *number1* or *number2*.

Example

```
REM Prints the value of number1  
number1=12.26  
number2=27.45  
PRINT MIN(number1, number2)  
END
```

Associated

MAX

MODE7PLOTIMAGE

Purpose

Experimental

Syntax

Not telling yet!

Description

Great fun

Example

Erm... early Nintendo kart based racing games !

Associated



MODE7PLOTSPRITE

Purpose

Experimental

Syntax

Not telling yet!

Description

Great fun

Example

Erm... early Nintendo kart based racing games !

Associated



MOUSEOFF

Purpose

Make the mouse cursor invisible.

Syntax

MOUSEOFF

Description

Make the mouse cursor invisible within the FUZE BASIC window. This is the default value.

Example

```
CLS
MOUSEON
LOOP
    GETMOUSE(x,y,b)
    LINETO(x,y)
    UPDATE
    REM LOOP colour if left button pressed
    IF b & 1 THEN
        COLOUR = COLOUR MOD 16 + 1
    ENDIF
    REM Exit if right button pressed
REPEAT UNTIL b & 4
MOUSEOFF
END
```

Associated

GETMOUSE, MOUSEON, MOUSEX, MOUSEY, SETMOUSE



MOUSEON

Purpose

Make the mouse cursor visible.

Syntax

MOUSEON

Description

Make the mouse cursor visible within the FUZE BASIC window. It is invisible by default.

Example

```
CLS
MOUSEON
LOOP
    GETMOUSE(x,y,b)
    LINETO(x,y)
    UPDATE
    REM LOOP colour if left button pressed
    IF b & 1 THEN
        COLOUR = COLOUR MOD 16 + 1
    ENDIF
    REM Exit if right button pressed
REPEAT UNTIL b & 4
MOUSEOFF
END
```

Associated

GETMOUSE, MOUSEOFF, MOUSEX, MOUSEY, SETMOUSE



MOUSEX

Purpose

To find the mouse X position

Syntax

value = MOUSEX

Description

Returns the X position of the current mouse location

Example

```
LOOP
PRINTAT(0,0); "Mouse X Position="; MOUSEX
;"      "
REPEAT
```

Associated

GETMOUSE, MOUSEOFF, MOUSEON, MOUSEY,
SETMOUSE

MOUSEY

Purpose

To find the mouse Y position

Syntax

value = MOUSEY

Description

Returns the Y position of the current mouse location

Example

```
LOOP
PRINTAT(0,0); "Mouse Y Position="; MOUSEY
;"      "
REPEAT
```

Associated

GETMOUSE, MOUSEOFF, MOUSEON, MOUSEX,
SETMOUSE

MOVE

Purpose

Move the graphics turtle forward

Syntax

MOVE(*distance*)

Description

This causes the virtual graphics turtle to move forwards *distance* in screen pixels. A line will be drawn if the pen is down.

Example

```
CLS
COLOUR=RED
MOVE(50)
DEG
LEFT(90)
MOVE(50)
PENDOWN
FOR I = 1 TO 4 LOOP
    LEFT(90)
    MOVE(100)
REPEAT
UPDATE
END
```

Associated

LEFT, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE



MOVETO

Purpose

Move the graphics turtle to a point on the screen.

Syntax

MOVETO(*xpos,ypos*)

Description

This moves the virtual graphics turtle to the absolute location (*xpos,ypos*). A line will be drawn if the pen is down.

Example

```
REM Draw a spiral in the centre of the
screen
CLS
COLOUR=RED
PENUP
MOVETO(GWIDTH/2,GHEIGHT/2)
PENDOWN
FOR I=2 TO GWIDTH LOOP
    MOVE(I)
    RIGHT(30)
REPEAT
UPDATE
END
```

Associated

LEFT, MOVE, PENDOWN, PENUP, RIGHT, TANGLE



NEWSPRITE

Purpose

Create a new sprite.

Syntax

index=NEWSPRITE(*count*)

Description

This returns an index (or handle) to the internal sprite data. You need to use the index returned in all future sprite handling functions/procedures. The *count* argument specifies the number of different versions of the sprite.

Example overleaf...

Example

```
CLS
COLOUR=YELLOW
CIRCLE(100,100,50,TRUE)
SAVEREGION("pac1.bmp",50,50,101,101)
COLOUR=BLACK
TRIANGLE(100,100,150,125,150,75,TRUE)
SAVEREGION("pac2.bmp",50,50,101,101)
TRIANGLE(100,100,150,150,150,50,TRUE)
SAVEREGION("pac3.bmp",50,50,101,101)
CLS
pacman=NEWSPRITE(3)
LOADSPRITE("pac1.bmp",pacman,0)
LOADSPRITE("pac2.bmp",pacman,1)
LOADSPRITE("pac3.bmp",pacman,2)
FOR X=1 TO GWIDTH STEP 25 LOOP
  FOR S=0 TO 2 LOOP
    PLOTSPRITE(pacman,X,GHEIGHT/2,S)
    UPDATE
    WAIT(.1)
  REPEAT
REPEAT
HIDESPRITE(pacman)
END
```

Associated

GETSPRITEH, GETSPRITEW, HIDESPRITE, LOADSPRITE, PLOTSPRITE, SETSPRITETRANS, SPRITECOLLIDE, SPRITECOLLIDEP

NUMFORMAT

Purpose

Control how numbers are formatted.

Syntax

NUMFORMAT(*width*,*decimals*)

Description

You can affect the way numbers are printed using the NUMFORMAT procedure. This takes 2 arguments, the *width* specifying the total number of characters to print and the *decimals* the number of characters after the decimal point. Numbers printed this way are right-justified with leading spaces inserted if required. NUMFORMAT (0,0) restores the output to the general purpose format used by default.

Example

```
NUMFORMAT(6,4)
REM Prints 3.1416
PRINT PI
NUMFORMAT(0,0)
REM Prints 3.141592654
PRINT PI
END
```

Associated

PRINT, PRINTAT



OPEN

Purpose

Open a file for read or write.

Syntax

handle=OPEN(*filename*\$)

Description

The OPEN function opens a file and makes it available for reading or writing and returns the numeric *handle* associated with the file. The file is created if it doesn't exist, or if it does exist the file pointer is positioned at the start of the file.

Example

```
handle = OPEN("testfile.txt")
PRINT# handle, "Colin"
PRINT# handle, 47
CLOSE(handle)
handle = OPEN("testfile.txt")
INPUT# handle, Name$
INPUT# handle, Age
CLOSE(handle)
PRINT "Name: " + Name$
PRINT "Age: "; Age
END
```

Associated

CLOSE, EOF, FFW, INPUT#, PRINT#, REWIND, SEEK



ORIGIN

Purpose

Move the graphics origin.

Syntax

`ORIGIN(xpos,ypos)`

Description

This changes the graphics origin for the Cartesian plotting procedures. The *xpos*, *ypos* coordinates are always absolute coordinates with (0,0) being bottom left (the default).

Example

```
CLS
COLOUR=yellow
REM Move the origin to the screen centre
ORIGIN(GWIDTH/2, GHEIGHT/2)
PLOT(-100,-100)
LINETO(-100,100)
LINETO(100,100)
LINETO(100,-100)
LINETO(-100,-100)
UPDATE
ORIGIN(0,0)
END
```

Associated

[GHEIGHT](#), [GWIDTH](#)



PAPER

Purpose

Set/Read the current text background colour.

Syntax

backgroundcolour=*PAPER*

PAPER=*backgroundcolour*

Description

Set/Read the current text background (paper) colour. Clear screen (CLS) will set the entire background to this colour.

Example

```
PRINT "Text background colour "; PAPER
PAPER=RED
PRINT "Text background colour "; PAPER
PAPER=7
PRINT "Text background colour "; PAPER
END
```

Associated

[INK](#), [PAPERON](#), [PAPEROFF](#)



PAPEROFF

Purpose

Switches the text background colour off.

Syntax

PAPEROFF

Description

This function switches the background text colour off. It can be turned on or off so that the text background does not obscure whatever is behind it.

Example

LOOP

INK = Orange

PaperOn

fontSize (RND (10) + 1)

printAt (RND(tWidth), RND(tHeight - 1)); "ON";

WAIT(.3)

INK = Yellow

PaperOff

fontSize (RND (10) + 1)

printAt (RND(tWidth), RND(tHeight - 1)); "OFF";

WAIT(.3)

REPEAT

Associated

INK, PAPER, PAPERON



PAPERON

Purpose

Sets text background colour to display

Syntax

PAPERON

Description

This function switches the background text colour on. It can be turned on or off so that the text background does not obscure whatever is behind it.

Example

LOOP

INK = Orange

PaperOn

fontSize (RND (10) + 1)

printAt (RND(tWidth), RND(tHeight - 1)); "ON";

WAIT(.3)

INK = Yellow

PaperOff

fontSize (RND (10) + 1)

printAt (RND(tWidth), RND(tHeight - 1)); "OFF";

WAIT(.3)

REPEAT

Associated

INK, PAPER, PAPEROFF



PAUSECHAN

Purpose

Pause the playing of a sound sample.

Syntax

PAUSECHAN(*handle*)

Description

This function pauses the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE*. The sample can be resumed where it left off using *RESUMECHAN*

Example

```
channel=0
volume=70
SETCHANVOL(channel,volume)
intro=LOADSAMPLE("pacman_intro.wav")
PLAYSAMPLE(intro,channel,0)
WAIT(3)
PAUSECHAN(intro)
Wait(2)
RESUMECHAN(intro)
END
```

Associated

[LOADSAMPLE](#), [PLAYSAMPLE](#), [RESUMECHAN](#),
[SETCHANVOL](#), [STOPCHAN](#)



PAUSEMUSIC

Purpose

Pause a playing music file

Syntax

PAUSEMUSIC

Description

Pauses a playing music track which can then be restarted using *RESUMEMUSIC*.

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
PAUSEMUSIC
```

Associated

[LOADMUSIC](#), [RESUMEMUSIC](#), [SETMUSICVOL](#), [STOPMUSIC](#)



PENDOWN

Purpose

Start drawing using the graphics turtle.

Syntax

PENDOWN

Description

This lowers the “pen” that the virtual graphics turtle is using to draw with. Nothing will be drawn until you execute this procedure.

Example

```
REM Draw a spiral in the centre of the
screen
CLS
COLOUR=RED
PENUP
MOVETO(GWIDTH/2,GHEIGHT/2)
PENDOWN
FOR I=2 TO GWIDTH LOOP
    MOVE(I)
    RIGHT(30)
REPEAT
UPDATE
END
```

Associated

LEFT, MOVE, MOVETO, PENUP, RIGHT, TANGLE



PENUP

Purpose

Stop drawing using the graphics turtle.

Syntax

PENUP

Description

This lifts the “pen” that the virtual graphics turtle uses to draw. You can move the turtle without drawing while the pen is up.

Example

```
REM Draw a spiral in the centre of the
screen
CLS
COLOUR=RED
PENUP
MOVETO(GWIDTH/2,GHEIGHT/2)
PENDOWN
FOR I=2 TO GWIDTH LOOP
    MOVE(I)
    RIGHT(30)
REPEAT
UPDATE
END
```

Associated

LEFT, MOVE, MOVETO, PENDOWN, RIGHT, TANGLE



PI / PI2

Purpose

Returns the value of the constant Pi and Pi divided by 2

Syntax

valueofpi=PI / *valueofpi*=PI divided by 2

Description

Returns an approximation of the value of the constant pi which is the ratio of a circle's circumference to its diameter (approximately 3.141592654) which is widely used in mathematics, specifically trigonometry and geometry.

Example

```
PRINT FN AreaOfCircle(12)
END
DEF FN AreaOfCircle(withRadius)
  LOCAL result
  result=PI*withRadius*withRadius
= result
```

PINMODE

Purpose

Configure the mode of a pin on the Pi's GPIO.

Syntax

PINMODE(*pinno*,*pinmode*)

Description

Configures the mode of a pin on the Pi's GPIO. It takes an argument which specifies the mode of the pin - input, output or PWM output. The modes are:

- 0 pinINPUT
- 1 pinOUTPUT
- 2 pinPWM

Example

```
REM Set pin 12 to input
PINMODE(12,1)
REM Wait for button to be pushed
UNTIL DIGITALREAD(12) LOOP
REPEAT
PRINT "Button Pushed"
END
```

Associated

DIGITALREAD, DIGITALWRITE, PWMWRITE,
SOFTPWMWRITE

PLAYMUSIC

Purpose

Start playing a music track.

Syntax

PLAYMUSIC(*handle*,*repeats*)

Description

This function plays a music track previously loaded using the *LOADMUSIC* function which returns the *handle*. The *repeats* are the number of times to play the track.

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
END
```

Associated

[LOADMUSIC](#), [PAUSEMUSIC](#), [RESUMEMUSIC](#),
[SETMUSICVOL](#), [STOPMUSIC](#)

PLAYSAMPLE

Purpose

Start playing a sound sample.

Syntax

PLAYSAMPLE(*handle*,*channel*,*loops*)

Description

This function plays a sound previously loaded using the *LOADSAMPLE* function which returns the *handle*. The *channel* is 0, 1, 2 or 3 which lets you play up to 4 concurrent samples. The *loops* parameter is different to the *repeats* one in the *PLAYMUSIC* function. Here it means the number of times to loop the sample – zero means no loops which means play it ONCE.

Example

```
channel=0
volume=70
SETCHANVOL(channel,volume)
intro=LOADSAMPLE("pacman_intro.wav")
PLAYSAMPLE(intro,channel,0)
WAIT(4.5)
END
```

Associated

[LOADSAMPLE](#), [PAUSECHAN](#), [RESUMECHAN](#), [SETCHANVOL](#),
[STOPCHAN](#)

PLOT

Purpose

Draw a single point on the screen.

Syntax

PLOT(*xpos,ypos*)

Description

This plots a single pixel at screen location (*xpos,ypos*) in the selected graphics mode in the selected colour. Note that (0,0) is bottom left by default.

Example

```
CLS
LOOP
  IF INKEY<>-1 THEN BREAK
  xpos=RND(GWIDTH)
  ypos=RND(GHEIGHT)
  COLOUR=RND(16)
  PLOT(xpos, ypos)
  UPDATE
REPEAT
END
```

Associated

LINE, LINETO, MOVETO, RECT, TRIANGLE, CIRCLE, ELLIPSE, COLOUR, RGBCOLOUR



PLOTIMAGE

Purpose

Display a loaded image on the screen.

Syntax

PLOTIMAGE(*handle,xpos,ypos*)

Description

Plot an image previously loaded using LOADIMAGE (using the *handle* returned by LOADIMAGE) on the screen at coordinates (*xpos,ypos*).

Example

```
COLOUR=RED
RECT(0,0,50,50,TRUE)
COLOUR=WHITE
LINE(0,50,50,50)
LINE(0,25,50,25)
LINE(0,25,0,50)
LINE(50,25,50,50)
LINE(25,0,25,25)
LINE(0,0,50,0)
SAVEREGION("bricks.bmp",0,0,50,50)
handle=LOADIMAGE("bricks.bmp")
FOR X=0 TO GWIDTH STEP 50 LOOP
  FOR Y=0 TO GHEIGHT STEP 50 LOOP
    PLOTIMAGE(handle,X,Y)
  REPEAT
REPEAT
UPDATE
END
```

Associated

GETIMAGEH, GETIMAGEW, LOADIMAGE, FREEIMAGE



PLOTSPRITE

Purpose

Draw a sprite on the screen.

Syntax

PLOTSPRITE(*index*,*xpos*,*ypos*,*subindex*)

Description

This plots the sprite *index* and version *subindex* at the coordinates (*xpos*, *ypos*). The coordinates specify the bottom-left corner of the bounding rectangle of the sprite.

Example

```
CLS
REM Create a new sprite with 1 version
SpriteIndex=NEWSPRITE(1)
REM Load a sprite from a file
fuzelogo$="/usr/share/fuze/logo.bmp"
LOADSPRITE(fuzelogo$,SpriteIndex,0)
REM Draw the sprite on the screen
PLOTSPRITE(SpriteIndex,200,200,0)
UPDATE
END
```

Associated

GETSPRITEH, GETSPRITEW, HIDESPRITE, LOADSPRITE,
NEWSPRITE, SETSPRITEALPHA, SETSPRITEANGLE,
SETSPRITEFLIP, SETSPRITESIZE, SETSPRITETRANS,
SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT



POLYEND

Purpose

Draw the filled polygon started by PolyStart.

Syntax

POLYEND

Description

This marks the end of drawing a polygon. When this is called, the stored points will be plotted on the screen and the polygon will be filled.

Example

```
CLS
PROC Hexagon(200,200,50,Red)
UPDATE
END
DEF Proc Hexagon(x,y,l,c)
    COLOUR=c
    POLYSTART
    POLYPLOT(x+l,y)
    POLYPLOT(x+l/2,y-l*SQRT(3/2))
    POLYPLOT(x-l/2,y-l*SQRT(3/2))
    POLYPLOT(x-l,y)
    POLYPLOT(x-l/2,y+l*SQRT(3/2))
    POLYPLOT(x+l/2,y+l*SQRT(3/2))
    POLYEND
ENDPROC
```

Associated

POLYPLOT, POLYSTART



POLYPLOT

Purpose

Add a point to a filled polygon.

Syntax

POLYPLOT(*xpos,ypos*)

Description

This remembers the given *xpos,ypos* coordinates as part of a filled polygon. Nothing is actually drawn on the screen until the PolyEnd instruction is executed. Polygons can have a maximum of 64 points.

Example

```
CLS
PROC Hexagon(200,200,50,Red)
UPDATE
END
DEF Proc Hexagon(x,y,l,c)
  COLOUR=c
  POLYSTART
  POLYPLOT(x+l,y)
  POLYPLOT(x+l/2,y-l*SQRT(3/2))
  POLYPLOT(x-l/2,y-l*SQRT(3/2))
  POLYPLOT(x-l,y)
  POLYPLOT(x-l/2,y+l*SQRT(3/2))
  POLYPLOT(x+l/2,y+l*SQRT(3/2))
  POLYEND
ENDPROC
```

Associated

POLYEND, POLYSTART



POLYSTART

Purpose

Start drawing a filled polygon.

Syntax

POLYSTART

Description

This marks the start of drawing a filled polygon.

Example

```
CLS
PROC Hexagon(200,200,50,Red)
UPDATE
END
DEF Proc Hexagon(x,y,l,c)
  COLOUR=c
  POLYSTART
  POLYPLOT(x+l,y)
  POLYPLOT(x+l/2,y-l*SQRT(3/2))
  POLYPLOT(x-l/2,y-l*SQRT(3/2))
  POLYPLOT(x-l,y)
  POLYPLOT(x-l/2,y+l*SQRT(3/2))
  POLYPLOT(x+l/2,y+l*SQRT(3/2))
  POLYEND
ENDPROC
```

Associated

POLYEND, POLYPLOT



PLOTTEXT

Purpose

Display text using graphic coordinates.

Syntax

```
PLOTTEXT("text", xpos, ypos)
```

Description

The PRINT command uses the cursor coordinates to display text whereas PLOTTEXT can position text at a specified pixel location

Example

```
LOOP
  INK = RND (30)
  fontSize (RND (10) + 1)
  plotText("HELLO", RND(gWidth), RND(gHeight-1))
  UPDATE
REPEAT
```

Associated

[PRINT](#), [PRINTAT](#)

PRINT

Purpose

Output text to the screen

Syntax

```
PRINT {text}{;text}
```

Description

Outputting text to the screen is done via the PRINT command. The PRINT command is quite versatile and will print any combination of numbers and strings separated by the semi-colon (;). A trailing semi-colon will suppress the printing of a new line.

Example

```
PRINT "Hello Colin"
PRINT "Hello ";
name$="Colin"
PRINT name$
PRINT "Hello "; name$
END
```

Associated

[NUMFORMAT](#), [PRINTAT](#)

PRINT#

Purpose

Print data to a file.

Syntax

PRINT# *handle,data*

Description

The PRINT# instruction acts just like the regular PRINT instruction except that it sends data to the file identified by the supplied file-handle rather than to the screen. Numbers are formatted according to the settings of NUMFORMAT. It is strongly recommended to only print one item per line if you are going to read those items back into a FUZE BASIC program again.

Example

```
handle = OPEN("testfile.txt")
PRINT# handle, "Hello World"
CLOSE (handle)
END
```

Associated

CLOSE, EOF, FWD, INPUT#, OPEN, REWIND, SEEK

PRINTAT

Purpose

Set the text cursor position and print

Syntax

PRINTAT(*x, y*); "*text*"

Description

Use to position the text cursor at the specified location and print. Useful for laying out text and or variables at any preferred screen location.

Example

```
name$ = "Sam"
age = 10
PRINTAT( 0, 5 ); "My name is "; name$
PRINTAT( 5, 10 ); "I am "; age ; " years
old"
END
```

Associated

CHR\$, HTAB, HVTAB, INK,
PAPER, PRINT, PRINT, THEIGHT, TWIDTH, VTAB

PROC

Purpose

Call a user defined procedure.

Syntax

PROC *name*({*argument*}{*argument*})

Description

Calls the specified user defined procedure called *name* with the specified *arguments*. Once the procedure has been executed control returns to the command following.

Example

```
CLS
LOOP
  x=RND(TWIDTH)
  y=RND(THEIGHT)
  c=RND(15)
  Text$="Blossom"
  PROC text(text$, x, y, c)
REPEAT
END
DEF PROC text(text$, x, y, c)
  INK=c
  PRINTAT(x,y); text$
ENDPROC
```

Associated

DEF PROC, ENDPROC



PWMWRITE

Purpose

Output a PWM waveform on the selected pin.

Syntax

PWMWRITE(*pinno*,*pinvalue*)

Description

This procedure outputs a PWM waveform on the selected pin. The pin must be configured for PWM mode beforehand. The value set should be between 0 and 100.

Example

```
REM Set pin 1 to PWM output mode
PINMODE(1,2)
PWMWRITE (1,50)
END
```

Associated

DIGITALREAD, DIGITALWRITE, PINMODE



RAD

Purpose

Set angle units to radians.

Syntax

RAD

Description

Switches the internal angle system to radians. There are $2 * \pi$ radians in a full circle.

Example

```
REM Draw an ellipse in the screen centre
CLS
RAD
FOR Angle=0 TO 2 * PI STEP 0.01 LOOP
    Xpos=100*COS(Angle)+GWIDTH / 2
    Ypos=50*SIN(Angle)+GHEIGHT / 2
    PLOT(Xpos,Ypos)
REPEAT
END
```

Associated

CLOCK, DEG



READ

Purpose

Read data into program variables.

Syntax

READ *variable* {*,variable*}

Description

To get data into your program variables, we use the READ instruction. We can read one, or many items of data at a time.

Example

```
REM Load the name of the days of the
REM week into a string array
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday"
DATA "Sunday"
DIM DaysOfWeek$(7)
FOR Day = 1 TO 7 LOOP
    READ DaysOfWeek$(Day)
REPEAT
PRINT "The third day of the week is ";
PRINT DaysOfWeek$(3)
END
```

Associated

DATA, RESTORE



RECT

Purpose

Draw a rectangle on the screen.

Syntax

RECT(*xpos,ypos,width,height,fill*)

Description

Draws a rectangle at position (*xpos,ypos*) with *width* and *height*. The final parameter, *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Example

CLS

LOOP

COLOUR=RND(16)

x=RND(GWIDTH)

y=RND(GHEIGHT)

w=RND(GWIDTH / 4)

h=RND(GHEIGHT / 4)

f=RND(2)

RECT(x,y,w,h,f)

UPDATE

IF INKEY<>-1 THEN BREAK

REPEAT

END

Associated

CIRCLE, ELLIPSE, TRIANGLE



REPEAT UNTIL

Purpose

Loop until the specified condition is met.

Syntax

LOOP

{*statements*}

REPEAT UNTIL *condition*

Description

Execute the *statements* one or more times until the *condition* is TRUE (not 0).

Example

Number=INT(RND(10)) + 1

Guess=0

REM Guessing Game

PRINT "Guess a Number Between 1 and 10"

LOOP

INPUT "Enter Your Guess: ", Guess

IF (Number<>Guess) THEN

PRINT "Incorrect Guess Again"

ENDIF

REPEAT UNTIL Number = Guess

PRINT "You are Correct!"

END

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT,



RESTORE

Purpose

Reset the DATA pointer.

Syntax

RESTORE

Description

With no *lineno* specified resets the READ command to the very first DATA statement in the program.

Example

```
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday",
"Sunday"
FOR Day = 1 TO 3 LOOP
    READ DayOfWeek$
REPEAT
PRINT DayOfWeek$
RESTORE
FOR Day = 1 TO 4 LOOP
    READ DayOfWeek$
REPEAT
PRINT DayOfWeek$
END
```

Associated

DATA, READ



RESUMECHAN

Purpose

Resume the playing of a sound sample.

Syntax

RESUMECHAN(*handle*)

Description

This function resumes the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE* and paused using *PAUSECHAN*.

Example

```
channel=0
volume=70
SETCHANVOL(channel,volume)
intro=LOADSAMPLE("pacman_intro.wav")
PLAYSAMPLE(intro,channel,0)
WAIT(3)
PAUSECHAN(intro)
WAIT(2)
RESUMECHAN(intro)
END
```

Associated

LOADSAMPLE, PAUSECHAN, PLAYSAMPLE, SETCHANVOL, STOPCHAN



RESUMEMUSIC

Purpose

Resumes music playing after it has been paused.

Syntax

RESUMEMUSIC

Description

Resumes the playing of a music track previously paused using PAUSEMUSIC.

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
PAUSEMUSIC
WAIT(1)
RESUMEMUSIC
```

Associated

[LOADMUSIC](#), [PAUSEMUSIC](#), [SETMUSICVOL](#), [STOPMUSIC](#)

REWIND

Purpose

Move the file pointer to the start of a file.

Syntax

REWIND(*handle*)

Description

Move the file pointer to the start of the file specified by *handle*.

Example

```
handle=OPEN ("rewindtest.txt")
PRINT# handle,"First Record"
PRINT# handle,"Second Record"
CLOSE(handle)
handle=OPEN("rewindtest.txt")
INPUT# handle, record$
PRINT record$
REWIND(handle)
REM reads the first record again
INPUT# handle, record$
PRINT record$
CLOSE (handle)
END
```

Associated

[CLOSE](#), [EOF](#), [FFWD](#), [INPUT#](#), [OPEN](#), [PRINT#](#), [SEEK](#)

RGBCOLOUR

Purpose

Set the current graphical plot colour to an RGB (Red,Green, Blue) value.

Syntax

RGBCOLOUR(*red,green,blue*)

Description

This sets the current graphical plot colour to an RGB (Red,Green, Blue) value. The values should be from 0 to 255.

Example

```
CLS
PRINT "Draw Spectrum"
FOR v = 0 TO 255 LOOP
  RGBCOLOUR(255,v,0)
  LINE(v,300,v,400)
  RGBCOLOUR(v,255,0)
  LINE(511-v,300,511-v,400)
  RGBCOLOUR(0,255-v,v)
  LINE(512+v,300,512+v,400)
  RGBCOLOUR(0,v,255)
  LINE(768+v,300,768+v,400)
REPEAT
UPDATE
END
```

Associated

COLOUR, INK, PAPER



RIGHT

Purpose

Turns the turtle to the right (clockwise) by the given angle.

Syntax

RIGHT(*angle*)

Description

Turns the virtual graphics turtle to the right (clockwise) by the given *angle* in the current angle units.

Example

```
CLS
PRINT "Draw Pink Hexagon"
PENUP
COLOUR = PINK
PENDOWN
FOR I = 1 TO 6 LOOP
  RIGHT(60)
  MOVE(100)
REPEAT
END
```

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, TANGLE



RIGHT\$

Purpose

Return the specified rightmost number of a characters from a string.

Syntax

```
substring$=RIGHT$(string$,number)
```

Description

Returns a substring of *string\$* with *number* characters from the right (end) of the string. If number is greater than or equal to the length of *string\$* then the whole string is returned.

Example

```
String$="The quick brown fox"  
FOR I=1 TO 20 LOOP  
    PRINT RIGHT$(String$, I)  
REPEAT  
END
```

Associated

[LEFT\\$, MID\\$](#)

ROTATEIMAGE

Purpose

Return the specified rightmost number of a characters from a string.

Syntax

```
substring$=RIGHT$(string$,number)
```

Description

Returns a substring of *string\$* with *number* characters from the right (end) of the string. If number is greater than or equal to the length of *string\$* then the whole string is returned.

Example

```
image = loadImage ("screen.png")  
plotImage (image, 0, 0)  
UPDATE  
WAIT (2)  
CLS  
rotateImage (image, 90)  
plotImage (image, 0, 0)  
UPDATE  
WAIT (2)  
END
```

Associated

[LOADIMAGE, SCALEIMAGE](#)

RND

Purpose

Generate a random number in a given range.

Syntax

random=RND(*range*)

Description

This function returns a random number based on the value of *range*. If *range* is zero, then the last random number generated is returned, if *range* is 1, then a random number from 0 to 1 is returned, otherwise a random number from 0 up to, but not including *range* is returned.

Example

```
DiceRoll=RND(6)+1
PRINT "Dice Roll: "; DiceRoll
CoinToss=RND(2)
IF CoinToss=0 THEN
    PRINT "Heads"
ELSE
    PRINT "Tails"
ENDIF
END
```

Associated

SEED



SAVEREGION

Purpose

Save a snapshot of an area of the screen to an image file.

Syntax

SAVEREGION(*file\$*,*xpos*,*ypos*,*width*,*height*)

Description

Takes a snapshot of an area of the current screen specified by the rectangle with bottom left at coordinates (*xpos*,*ypos*) of specified *width* and *height*, and saves it to the file named *file\$* in a bitmap (.bmp) format.

SAVEREGION Example

```
CLS
COLOUR=YELLOW
CIRCLE (100,100,50,TRUE)
COLOUR=BLACK
TRIANGLE(100,100,150,150,150,50,TRUE)
TRIANGLE(100,100,150,125,150,75,TRUE)
SAVEREGION ("pac1.bmp",50,50,101,101)
CLS
pacman=NEWSPRITE(1)
LOADSPRITE("pac1.bmp",pacman,0)
FOR X=1 TO GWIDTH STEP 25 LOOP
PLOTSPRITE(pacman,X,GHEIGHT/2,0)
UPDATE
WAIT(.1)
REPEAT
```

Associated

SAVESCREEN



SAVESCREEN

Purpose

Save a snapshot of the screen to an image file.

Syntax

SAVESCREEN(*filename*\$)

Description

This takes a snapshot of the current screen and saves it to the filename given in a bitmap (.bmp) format file.

Example

```
CLS
PRINT "Draw Spectrum"
FOR v = 0 TO 255 LOOP
  RGBCOLOUR(255,v,0)
  LINE(v,300,v,400)
  RGBCOLOUR(v,255,0)
  LINE(511-v,300,511-v,400)
  RGBCOLOUR(0,255-v,v)
  LINE(512+v,300,512+v,400)
  RGBCOLOUR(0,v,255)
  LINE(768+v,300,768+v,400)
REPEAT
UPDATE
SAVESCREEN("screenshot.bmp")
END
```

Associated

SAVEREGION



SCALEIMAGE

Purpose

Resize a loaded image.

Syntax

SCALEIMAGE(*handle*, *percent*)

Description

Scales a preloaded image by a specified percentage.

Example

```
image = loadImage("screen.png")
plotImage (image, 0, 0)
UPDATE
WAIT (2)
CLS
scaleImage (image, 50)
plotImage (image, 0, 0)
UPDATE
WAIT (2)
END
```

Associated

LOADIMAGE, ROTATEIMAGE



SCANKEYBOARD

Purpose

Scan for a key pressed down.

Syntax

SCANKEYBOARD(*keycode*)

Description

Allows you to detect that any of the keys have been pressed (including special keys) and also to detect mutiple keys pressed at the same time. The *keycode* parameter indicates the key press to be scanned for e.g scanSpace is the space bar. See the end of this guide for a full list of SCANKEYBOARD keycodes.

Example

```
PRINT "Press Ctrl-Alt-Delete"
```

```
LOOP
```

```
    LCtrl = SCANKEYBOARD(scanLCtrl)
```

```
    LAlt  = SCANKEYBOARD(scanLAlt)
```

```
    Delete = SCANKEYBOARD(scanDelete)
```

```
    Reboot = LCtrl AND LAlt AND Delete
```

```
REPEAT UNTIL Reboot
```

```
PRINT "Rebooting..."
```

```
CLEARKEYBOARD
```

```
END
```

Associated

CLEARKEYBOARD, GET, INKEY, INPUT



SCLOSE

Purpose

Close an open serial port.

Syntax

SCLOSE(*handle*)

Description

This closes a serial port and frees up any resources used by it. It's not strictly necessary to do this when you end your program, but it is considered good practice.

Example

```
REM Read a character from a serial port
```

```
arduino=SOPEN("/dev/ttyUSB0", 115200)
```

```
char$=SGET$(arduino)
```

```
PRINT char$
```

```
SCLOSE(arduino)
```

```
END
```

Associated

SGET, SGET\$, SOPEN, SPUT, SPUT\$, SREADY



SCROLLEDOWN SCROLLLEFT SCROLLRIGHT SCROLLUP

Purpose

Scroll a region of the screen down.

Syntax

`SCROLLEDOWN(xpos,ypos,width,height,pixels)`

Description

Scroll the region of the screen specified by the rectangle at position (*xpos,ypos*) with dimensions *width* X *height* down by the specified number of *pixels*.

Example overleaf

Example

```
CLS
W=100 // Width
S=2 // Step Size
X=(GWIDTH-W)/2
Y=(GHEIGHT-W)/2
COLOUR=WHITE
RECT(X,Y,W,W,TRUE)
RECT(X+W,Y+W,W,W,TRUE)
RECT(X-2,Y-2,W*2+4,W*2+4,FALSE)
UPDATE
COLOUR=BLACK
FOR I=1 TO W STEP S LOOP
    SCROLLUP(X,Y,W,W*2,S)
    SCROLLEDOWN(X+W,Y,W,W*2,S)
    UPDATE
    WAIT(0.01)
REPEAT
FOR I = 1 TO W STEP S LOOP
    SCROLLRIGHT(X,Y+W,W*2,W,S)
    SCROLLLEFT(X,Y,W*2,W,S)
    WAIT(0.01)
UPDATE
REPEAT
END
```

Associated

[SCROLLLEFT](#), [SCROLLRIGHT](#), [SCROLLUP](#)

SEED

Purpose

Seed the random number generator.

Syntax

SEED=*value*

Description

This can be assigned to initialise the random number generator, or you can read it to find the current seed.

Example

```
SEED=10
PRINT RND(100)
SEED=10
REM Will print the same number
PRINT RND(100)
REM Will print a different number
PRINT RND(100)
END
```

Associated

[RND](#)

SEEK

Purpose

Move the file pointer to any place in the file.

Syntax

SEEK(*handle,offset*)

Description

The SEEK instruction moves the file pointer to any place in the file. It can even move the file pointer beyond the end of the file in which case the file is extended. The argument supplied to SEEK is an absolute number of bytes from the start of the file. If you are using random access files and want to access the 7th record in the file, then you need to multiply your record size by 7 to get the final location to seek to.

Example

```
handle=OPEN("TestFile.txt")
recSize = 20
FOR recNo=0 TO 10 LOOP
    record$ = "Record " + STR$(recNo)
    pad = recSize - LEN(record$)
    PRINT# handle,record$;SPACE$(pad)
REPEAT
REM read the 7th record
SEEK (handle,(recSize+1)*7)
INPUT# handle,record$
PRINT record$
CLOSE(handle)
END
```

Associated

[CLOSE](#), [EOF](#), [FFWD](#), [INPUT#](#), [OPEN](#), [PRINT#](#), [REWIND](#)



SENSEACCELX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT accelerometer.

Syntax

value=SENSEACCELX

value=SENSEACCELY

value=SENSEACCELZ

Description

The Raspberry Pi senseHAT has a number of built in sensors. The accelerometer can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Sense Accelerometer X="; senseAccelX
PRINT "Sense Accelerometer Y="; senseAccelY
PRINT "Sense Accelerometer Z="; senseAccelZ
REPEAT
END
```

Associated

[senseCompass](#), [senseGyro](#)

SENSECLS

Purpose

Sets all of the LEDs on the Raspberry Pi senseHAT to off.

Syntax

SENSECLS

Description

Sets an RGB value of 0, 0, 0 to all of the matrix LEDs thereby clearing the display.

Example

```
CLS
sensePlot(2,2)
WAIT (1)
senseCls
END
```

Associated

[sensePlot](#), [senseRect](#), [senseScroll](#), [senseHFlip](#), [senseVflip](#), [senseRGBcolour](#), [senseGetRGB](#), [senseLine](#)

SENSECOMPASSX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT compass.

Syntax

value=SENSECOMPASSX

value=SENSECOMPASSY

value=SENSECOMPASSZ

Description

The Raspberry Pi senseHAT has a number of built in sensors. The compass can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Sense Compass X="; senseCompassX
PRINT "Sense Compass Y="; senseCompassY
PRINT "Sense Compass Z="; senseCompassZ
REPEAT
END
```

Associated

[senseAccel](#), [senseGyro](#),

SENSEGETRGB

Purpose

Return the values used by a Raspberry Pi senseHAT LED

Syntax

SENSEGETRGB(*xpos*, *ypos*, *red*, *green*, *blue*)

Description

Returns the Red, Green and Blue values from a given LED at the specified matrix coordinates. It is possible to use this for collision detection in games using the senseHAT.

Example

```
CLS
senseRGBcolour (0, 0, 255)
sensePlot (0, 0)
SenseGetRGB (0, 0, R, G, B)
PRINT "Red="; R
PRINT "Green="; G
PRINT "Blue="; B
END
```

Associated

[senseCls](#), [sensePlot](#), [senseRect](#), [senseScroll](#), [senseHFlip](#),
[senseVflip](#), [senseRGBcolour](#), , [senseLine](#)

SENSEGYROX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT gyro.

Syntax

value=SENSEGYROX

value=SENSEGYROY

value=SENSEGYROZ

Description

The Raspberry Pi senseHAT has a number of built in sensors. The gyro can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Sense Gyro X="; senseGyroX
PRINT "Sense Gyro Y="; senseGyroY
PRINT "Sense Gyro Z="; senseGyroZ
REPEAT
END
```

Associated

[senseAccel](#), [senseCompass](#)

SENSEHEIGHT

Purpose

Returns the value of the Raspberry Pi senseHAT height sensor.

Syntax

value=SENSEHEIGHT

Description

The Raspberry Pi senseHAT has a number of built in sensors. Height above sea level can be accessed with this function. Note, this does not work well indoors.

Example

```
CLS
LOOP
PRINT "Height"; senseHeight
REPEAT
END
```

Associated

[senseHumidity](#), [sensePressure](#), [senseTemperature](#)

SENSEHFLIP

Purpose

Horizontally flips the Raspberry Pi senseHAT LED matrix.

Syntax

SENSEHFLIP

Description

Reverses (flips) the LED matrix display horizontally.

Example

```
CLS
SenseRGBcolour(255,0,0)
senseLine(0,0,0,7)
SenseRGBcolour(0,255,0)
senseLine(0,7,7,7)
LOOP
senseHflip
WAIT(1)
SenseVflip
WAIT(1)
END
```

Associated

[senseCls](#), [sensePlot](#), [senseRect](#), [senseScroll](#), [senseVflip](#),
[senseRGBcolour](#), [senseGetRGB](#), , [senseLine](#)



SENSEHUMIDITY

Purpose

Returns the value of the Raspberry Pi senseHAT humidity sensor.

Syntax

value=SENSEHUMIDITY

Description

The Raspberry Pi senseHAT has a number of built in sensors. Humidity can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Humidity"; senseHumidity
REPEAT
END
```

Associated

[senseHeight](#), [sensePressure](#), [senseTemperature](#)



SENSELINE

Purpose

Lights a line on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSELINE(*x1*, *y1*, *x2*, *y2*)

Description

Sets the RGB values, defined by `senseRGBcolour`, to a line of LEDs on the Raspberry Pi senseHAT LED matrix. The line is displayed from *x1*, *y1* to *x2*, *y2*.

Example

```
CLS
SenseRGBcolour(255,0,0)
senseLine(0,0,0,7)
SenseRGBcolour(0,255,0)
senseLine(0,7,7,7)
LOOP
senseHflip
WAIT(1)
SenseVflip
WAIT(1)
END
```

Associated

`senseCls`, `sensePlot`, `senseRect`, `senseScroll`, `senseVflip`,
`senseRGBcolour`, `senseGetRGB`



SENSEPLOT

Purpose

Lights an LED on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSEPLOT(*xpos*, *ypos*)

Description

Sets the RGB values, defined by `senseRGBcolour`, to a single LED on the Raspberry Pi senseHAT LED matrix.

Example

```
CLS
SenseRGBcolour(255,0,0)
sensePlot(0,0)
SenseRGBcolour(0,255,0)
sensePlot(7,7)
END
```

Associated

`senseCls`, `senseHflip`, `senseRect`, `senseScroll`, `senseVflip`,
`senseRGBcolour`, `senseGetRGB`, `senseLine`



SENSEPRESSURE

Purpose

Returns the value of the Raspberry Pi senseHAT air pressure sensor.

Syntax

value=SENSEPRESSURE

Description

The Raspberry Pi senseHAT has a number of built in sensors. Air pressure can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Pressure"; sensePressure
REPEAT
END
```

Associated

[senseHumidity](#), [senseHeight](#), [senseTemperature](#)

SENSERECT

Purpose

Lights a rectangle on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSERECT(*xpos*, *ypos*, *width*, *height*, *fill*)

Description

Sets the RGB values, defined by [senseRGBcolour](#), to a rectangle of LEDs on the Raspberry Pi senseHAT LED matrix. The rectangle is displayed from *xpos*, *ypos* with a width and height as specified. Fill can be either 0 for an outline or 1 for filled in.

Example

```
CLS
SenseRGBcolour(255,0,0)
senseRect(0,0,7,7,1)
SenseRGBcolour(0,255,0)
senseRect(0,0,7,7,0)
END
```

Associated

[senseCls](#), [sensePlot](#), [senseLine](#), [senseScroll](#), [senseVflip](#), [senseRGBcolour](#), [senseGetRGB](#)

SENSEGBCOLOUR

Purpose

Set the Red, Green and Blue values used by a Raspberry Pi senseHAT LED.

Syntax

SENSEGBCOLOUR(*red, green, blue*)

Description

Sets the Red, Green and Blue values to be used by the senseHAT FUZE BASIC drawing commands.

Example

```
CLS
senseRGBcolour (255, 0, 0)
sensePlot (0, 0)
senseRGBcolour (0, 255, 0)
sensePlot (3, 3)
senseRGBcolour (0, 0, 255)
sensePlot (7, 7)
END
```

Associated

[senseCls](#), [sensePlot](#), [senseRect](#), [senseScroll](#), [senseHFlip](#),
[senseVflip](#), [senseLine](#), [senseGetRGB](#)



SENSESCROLL

Purpose

Scrolls the Raspberry Pi senseHAT LED matrix.

Syntax

SENSESCROLL(*direction, direction*)

Description

Shifts the Raspberry Pi senseHAT LED matrix in the specified direction by the number of pixels indicated.

Example

```
senseCLS
sensePlot (2, 2)
sensePlot (2, 3)
LOOP
SenseScroll (0, 2) // two up
WAIT(0.1)
SenseScroll (2, 0) // two right
WAIT(0.1)
SenseScroll (0, -2) // two down
WAIT(0.1)
SenseScroll (-2, 0) // two left
WAIT(0.1)
REPEAT
END
```

Associated

[senseCls](#), [sensePlot](#), [senseRect](#), [senseRGBcolour](#), [senseHFlip](#),
[senseVflip](#), [senseLine](#), [senseGetRGB](#)



SENSETEMPERATURE

Purpose

Returns the value of the Raspberry Pi senseHAT heat sensor.

Syntax

value=SENSETEMPERATURE

Description

The Raspberry Pi senseHAT has a number of built in sensors. Temperature, in degrees, can be accessed with this function.

Example

```
CLS
LOOP
PRINT "Temperature"; senseTemperature
REPEAT
END
```

Associated

[senseHumidity](#), [senseHeight](#), [sensePressure](#)

SENSEVFLIP

Purpose

Vertically flips the Raspberry Pi senseHAT LED matrix.

Syntax

SENSEHFLIP

Description

Reverses (flips) the LED matrix display vertically.

Example

```
CLS
SenseRGBcolour(255,0,0)
senseLine(0,0,0,7)
SenseRGBcolour(0,255,0)
senseLine(0,7,7,7)
LOOP
senseHflip
WAIT(1)
SenseVflip
WAIT(1)
END
```

Associated

[senseCls](#), [sensePlot](#), [senseRect](#), [senseScroll](#), [senseVflip](#),
[senseRGBcolour](#), [senseGetRGB](#), [senseLine](#)

SETCHANVOL

Purpose

Set the volume of a sound sample.

Syntax

SETCHANVOL(*channel*,*volume*)

Description

Sets the sound sample playback volume on the specified *channel* where *volume* is a percentage of the maximum (0-100)

Example

```
channel=0
chomp=LOADSAMPLE("pacman_chomp.wav")
FOR volume=10 TO 100 STEP 10 LOOP
    SETCHANVOL(channel,volume)
    PLAYSAMPLE(chomp,0,0)
    WAIT(1)
REPEAT
END
```

Associated

[LOADSAMPLE](#), [PAUSECHAN](#), [PLAYSAMPLE](#), [RESUMECHAN](#), [STOPCHAN](#)

SETMODE

Purpose

Set display width and height

Syntax

SETMODE(*width*, *height*)

Description

Sets the display width and height to the specified. It is generally sensible to use standard screen display sizes

Example

```
SETMODE( 1280, 720 )
PRINT "Hello World"
WAIT( 2 )
SETMODE( 640, 480 )
PRINT "Hello Another World"
END
```

Associated

[GHEIGHT](#), [GWIDTH](#)

SETMOUSE

Purpose

Move the mouse pointer to the specified point.

Syntax

SETMOUSE(*xpos,ypos*)

Description

Moves the mouse pointer to the screen coordinate (*xpos,ypos*)

Example

```
CLS
MOUSEON
COLOUR=WHITE
RECT(100,100,150,50,TRUE)
INK=BLACK
PAPER=WHITE
PRINTAT(7,38); "Click Me"
UPDATE
Clicked=FALSE
LOOP
  GETMOUSE(X,Y,Z)
  IF Z <> 0 THEN
    IF (X > 100 AND X < 250) THEN
      IF (Y > 100 AND Y < 150) THEN
        Clicked=TRUE
      ENDIF
    ENDIF
  ENDIF
REPEAT UNTIL Clicked
SETMOUSE(GWIDTH/2,GHEIGHT/2)
```

Associated

GETMOUSE, MOUSEOFF, MOUSEON, MOUSEX, MOUSEY



SETMUSICVOL

Purpose

Sets the music playback volume.

Syntax

SETMUSICVOL(*level*)

Description

Sets the music playback volume where *level* is a percentage of the maximum (0-100)

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
```

Associated

LOADMUSIC, PAUSEMUSIC, RESUMEMUSIC, STOPMUSIC



SETSPRITEALPHA

Purpose

Sets the transparency of a sprite

Syntax

SETSPRITEALPHA(*sprite*, *alpha*)

Description

Sets how transparent a sprite is. An alpha of 0 means it's invisible and 255 means it's completely opaque, or solid.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE( "/usr/share/fuze/logo.bmp",
pic, 0 )
FOR alpha = 0 TO 255 LOOP
SETSPRITEALPHA( pic, alpha )
PLOTSPRITE( pic, GWIDTH / 2, GHEIGHT / 2, 0
)
UPDATE
REPEAT
END
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITEANGLE, SETSPRITEFLIP,
SETSPRITEORIGIN, GETSPRITEX, GETSPRITEY



SETSPRITEANGLE

Purpose

Rotate a sprite to the given angle

Syntax

SETSPRITEANGLE(*sprite*, *angle*)

Description

Use to rotate the specified sprite to the given angle in degrees. 0 is default.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE( "/usr/share/fuze/logo.bmp",
pic, 0 )
FOR angle = 0 TO 360 LOOP
SETSPRITEANGLE( pic, angle )
PLOTSPRITE( pic, GWIDTH / 2, GHEIGHT / 2, 0
)
UPDATE
REPEAT
END
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITESIZE, SETSPRITEFLIP, SETSPRITEORIGIN,
GETSPRITEX, GETSPRITEY



SETSPRITEFLIP

Purpose

Mirror a sprite in the specified direction

Syntax

SETSPRITEFLIP(*sprite*, *flip*)

Description

Graphically mirrors (flips) the specified sprite.

- 0 Reset to default
- 1 mirrored vertically
- 2 mirrored horizontally
- 3 mirrored vertically & horizontally

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE("/usr/share/luze/logo.bmp",pic, 0)
PLOTSPRITE(pic, gWidth / 2, gHeight / 2, 0)
FOR a=3 TO 0 step -1 LOOP
  SETSPRITEFLIP( pic, a )
UPDATE
WAIT( 1 )
REPEAT
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEANGLE, SETSPRITESIZE, SETSPRITEORIGIN, GETSPRITE, GETSPRITE



SETSPRITEORIGIN

Purpose

Sets the anchor point of a sprite

Syntax

SETSPRITEorigin(*sprite*, *xpos*, *ypos*)

Description

Use to set the origin of a specified sprite. When plotting a sprite its default origin is bottom left. You can change this to any point on the sprite. This example sets it to the middle.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE( "logo.png", pic, 0 )
LOOP
  SETSPRITEORIGIN( pic, 0, 0 )
  Plotsprite( pic, 0, 0, 0)
Update
Wait(1)
MiddleX=GETSPRITEW(pic)/2
MiddleY=GETSPRITEH(pic)/2
SETSPRITEORIGIN( pic, MiddleX, MiddleY )
PLOTSPRITE( pic, 0, 0, 0)
UPDATE
REPEAT
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITESIZE, GETSPRITE, GETSPRITE



SETSPRITESIZE

Purpose

Change the size of a sprite

Syntax

SETSPRITESIZE(*sprite*, *size*)

Description

Sets the sprite to the specified *size* in percent. 100 is the default, therefore 50 is half the size and 300 is three times as big as the original.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE("/usr/share/fuze/logo.bmp",pic, 0)
FOR angle = 50 TO 200 LOOP
  SETSPRITESIZE( pic, size )
  PLOTSPRITE( pic, GWIDTH / 2, GHEIGHT / 2, 0)
UPDATE
REPEAT
END
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, SETSPRITEANGLE, SETSPRITEFLIP,
SETSPRITEORIGIN, GETSPRITEX, GETSPRITEY

SETSPRITETRANS

Purpose

Set a transparency colour for a given sprite and its sub-sprites.

Syntax

SETSPRITETRANS(*spriteId*, *Red*, *Green*, *Blue*)

Description

If you specify a transparency colour for a sprite then any pixels in the sprite that are this colour will be transparent i.e. They will show as the background colour. This allows a sprite to pass over a background image without blocking it out. You need to load the sprite files first before setting the transparency colour.

Example (overleaf)

SETSPRITETRANS Example

```
CLS
RGBCOLOUR(247, 247, 247)
RECT(50, 50, 101, 101, TRUE)
COLOUR = YELLOW
CIRCLE(100, 100, 50, TRUE)
SAVEREGION("s3.bmp", 50, 50, 101, 101)
CLS2
COLOUR = RED
MidY = GHEIGHT / 2
FOR X = 0 TO GWIDTH STEP 100 LOOP
    RECT(X, MidY - 50, 50, 200, TRUE)
REPEAT
COLOUR = Black
s1 = NEWSPRITE(1)
LOADSPRITE("s3.bmp", s1, 0)
SETSPRITETRANS(s1, 247, 247, 247)
FOR X = 0 TO gWidth STEP 10 LOOP
    PLOTSPRITE(s1, X, GHEIGHT / 2, 0)
    UPDATE
    WAIT (0.0005)
REPEAT
END
```

Associated

GETSPRITEH, GETSPRITEW, LOADSPRITE, NEWSPRITE,
PLOTSPRITE, RGBCOLOUR

SGET

Purpose

Read a byte from a serial port.

Syntax

byte=SGET(*handle*)

Description

Fetch a single byte of data from an open serial port and return the data as a number. This function will pause program execution for up to 5 seconds if no data is available. If there is still not data after 5 seconds, the function will return -1.

Example

```
REM Read a byte from a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
byte=SGET(arduino)
PRINT byte
SCLOSE(arduino)
END
```

Associated

SCLOSE, SGET\$, SOPEN, SPUT, SPUT\$, SREADY

SGET\$

Purpose

Read a character from a serial port.

Syntax

character=SGET\$(*handle*)

Description

Fetch a single byte of data from an open serial port and return the data as a single character string. This function will pause program execution for up to 5 seconds if no data is available. If there is still not data after 5 seconds, the function will return an empty string.

Example

```
REM Read a character from a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
char$=SGET$(arduino)
PRINT char$
SCLOSE(arduino)
END
```

Associated

SCLOSE, SGET, SOPEN, SPUT, SPUT\$, SREADY

SGN

Purpose

Returns the sign of the specified number.

Syntax

sign=SGN(*number*)

Description

Returns -1 if the number is negative, 1 otherwise. (Zero is considered positive)

Example

```
REM Prints 1
PRINT SGN(100)
PRINT SGN(0)
REM Print -1
PRINT SGN(-5)
END
```

Associated

ABS

SHOWKEYS

Purpose

List the function key definitions

Syntax

SHOWKEYS

Description

It is possible to set the 12 function keys at the top of the keyboard to user defined values. This command will show what the current definitions are. By default function key **F2** is defined to enter the **EDIT** command and **F3** the **RUN** command but it is possible to overwrite these.

Example

```
REM Set key F5 to clear screen
keyF5$ = "CLS\n"
SHOWKEYS
```

SIN

Purpose

Returns the sine of the given angle.

Syntax

sine=SIN(*angle*)

Description

Returns the sine of the argument *angle* in radians. This is the ratio of the side of a right angled triangle, that is opposite to the angle, to the hypotenuse (the longest side).

Example

```
REM Draw an ellipse in the screen centre
CLS
DEG
FOR Angle=0 TO 360 LOOP
    Xpos=100*COS(Angle)+GWIDTH/2
    Ypos=50*SIN(Angle)+GHEIGHT/2
    PLOT(Xpos,Ypos)
REPEAT
END
```

Associated

ASIN, ATAN, COS

SOFTPWMWRITE

Purpose

Synthesize Pulse Wave Modulation to the specified GPIO pin.

Syntax

`SOFTPWMWRITE(pinNo,value)`

Description

This enables you to simulate an analog output. For example instead of an LED being just on or off you can make it appear brighter or dimmer. The *pinNo* parameter is the number of the GPIO output pin. This must first be set to *pinSoftPwm*. The *value* parameter is a percentage (0-100).

Example

```
REM Connect an LED to GPIO pin 0
PINMODE (0, PINSOFTPWM)
FOR I=0 TO 100 LOOP
    SOFTPWMWRITE (0,I)
    WAIT(0.1)
REPEAT
FOR I=100 to 0 STEP -1 LOOP
    SOFTPWMWRITE (0,I)
    WAIT(0.1)
REPEAT
END
```

Associated

`PINMODE`



SOPEN

Purpose

Opens a serial device and makes it available for use.

Syntax

`handle=SOPEN($device,speed)`

Description

This opens a serial device and makes it available for our use. It takes the name of the serial port and the speed as an argument and returns a number (the handle) of the device. We can use this handle to reference the device and allow us to open several devices at once. The following baud rates are recognised: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 19200, 38400, 57600, 115200 and 230400, but do check your local PC and devices capabilities. The device is always opened with the data format set to 8 data bits, 1 stop bit and no parity. All handshaking is turned off.

Example

```
REM Read a byte from a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
byte=SGET(arduino)
SCLOSE(arduino)
END
```

Associated

`SCLOSE, SGET, SGET$, SPUT, SPUT$, SREADY`



SOUND

Purpose

Play a synthesised sound with the specified parameters (emulates BBC BASIC SOUND command).

Syntax

`sound(channel,amplitude,pitch,duration)`

Description

The *channel* is 0 to 3 with 0 being for white noise (static). The *amplitude* goes from 0 to -15 where 0 is silent and -15 is full volume (-7 being half volume and so on). The *duration* is in 20ths of a second, so 20 represents one second, 10 half a second and so on. The *pitch* values are taken from a table which can be found at the end of the manual but, middle C has a value of 53 and each note is 4 away from the next. Note that when used with a sound envelope *amplitude* is replaced by the envelope number.

Example

```
Channel=1
Volume=-10
FOR Note=1 TO 5 LOOP
    READ Frequency,Duration
    SOUND(Channel,Volume,Frequency,Duration)
REPEAT
END
DATA 89, 20, 97, 20, 81, 20, 33, 20, 61, 40
```

Associated

ENVELOPE, TONE



SPACE\$

Purpose

Returns a blank string of the specified length.

Syntax

`blankstring=SPACE$(number)`

Description

Returns a string of blank spaces *number* characters long.

Example

```
Blank$ = SPACE$(100)
FOR I=1 TO 20 LOOP
    PRINT Blank$
REPEAT
END
```



SPRITECOLLIDE

Purpose

Detect a sprite collision (fast bounding box)

Syntax

collision=SPRITECOLLIDE(*target*)

Description

Returns the sprite index of the first sprite that the sprite index *target* has collided with, or -1 if there is no collision. It only checks the current sprite location and this is only updated after a screen update LOOP so it is possible to call PLOTSPRITE() and have a sprite overlap but it not be detected until after the update has happened on screen.

Example

```
CLS
RGBCOLOUR(254,254,254)
RECT(50,50,101,101,TRUE)
COLOUR=YELLOW
CIRCLE(100,100,50,TRUE)
SAVEREGION("s1.bmp",50,50,101,101)
COLOUR=RED
CIRCLE(100,100,50,TRUE)
SAVEREGION("s2.bmp",50,50,101,101)
CLS2
s1=NEWSPRITE(1)
s2=NEWSPRITE(1)
LOADSPRITE("s1.bmp",s1,0)
LOADSPRITE("s2.bmp",s2,0)
FOR X=0 TO GWIDTH STEP 1 LOOP
    PLOTSPRITE(s1,X,200,0)
    PLOTSPRITE(s2,GWIDTH-X-100,200,0)
    IF SPRITECOLLIDE (s2) <> -1 THEN BREAK
    UPDATE
    WAIT(0.0005)
REPEAT
END
```

Associated

[GETSPRITEH](#), [GETSPRITEW](#), [HIDESPRITE](#), [LOADSPRITE](#),
[NEWSPRITE](#), [PLOTSPRITE](#), [SPRITECOLLIDEPP](#)

SPRITECOLLIDEPP

Purpose

Detect a sprite collision (pixel perfect)

Syntax

collision=SPRITECOLLIDEPP(*target,accuracy*)

Description

This is a slower but more accurate version of SpriteCollide. It first does do a simple bounding box test then checks row at a time. The *accuracy* parameter is how many pixels to skip both horizontally and vertically. This is from 1 to 16, where 1 is "perfect" and greater than 1 is less accurate, but faster. This will affect the amount of visible overlap you get before a collision is detected.

Example

```
CLS
RECT(50,50,101,101,TRUE)
COLOUR=YELLOW
CIRCLE(100,100,50,TRUE)
SAVEREGION("s1.bmp",50,50,101,101)
COLOUR=RED
CIRCLE(100,100,50,TRUE)
SAVEREGION("s2.bmp",50,50,101,101)
CLS2
s1=NEWSPRITE(1)
s2=NEWSPRITE(1)
LOADSPRITE("s1.bmp",s1,0)
LOADSPRITE("s2.bmp",s2,0)
FOR X=0 TO GWIDTH STEP 1 LOOP
    PLOTSPRITE(s1,X,200,0)
    PLOTSPRITE(s2,GWIDTH-X-100,200,0)
    IF SPRITECOLLIDEPP(s2,1) <> -1 THEN BREAK
UPDATE
REPEAT
END
```

Associated

[GETSPRITEH](#), [GETSPRITEW](#), [HIDESPRITE](#), [LOADSPRITE](#),
[NEWSPRITE](#), [PLOTSPRITE](#), [SPRITECOLLIDE](#)

SPRITEOUT

Purpose

Find out if a sprite is off screen

Syntax

result = SPRITEOUT(*sprite*)

Description

If the sprite is off screen then result is true otherwise it is false.

Example

```
pic = NEWSPRITE( 1 )
LOADSPRITE("/usr/share/fuze/logo.bmp",pic, 0)
PLOTSPRITE( pic, GWIDTH / 2, GHEIGHT / 2, 0)
WHILE NOT SPRITEOUT( pic ) LOOP
ADVANCESPRITE( pic, 2 )
UPDATE
REPEAT
PRINT "GONE!"
END
```

Associated

[HIDESPRITE](#), [PLOTSPRITE](#)

SPUT

Purpose

Send a byte to an open serial port.

Syntax

SPUT(*arduino*,*byte*)

Description

Send a single byte of data to an open serial port.

Example

```
REM Write a byte to a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
SPUT(arduino,52)
SPUT(arduino,50)
SCLOSE(arduino)
END
```

Associated

[SCLOSE](#), [SGET](#), [SGET\\$](#), [SOPEN](#), [SPUT\\$](#), [SREADY](#)

SPUT\$

Purpose

Send a character string to an open serial port.

Syntax

`SPUT$(arduino,string)`

Description

Send a string of characters of data to an open serial port.

Example

```
REM Write a byte to a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
SPUT$(arduino,"Hello")
SCLOSE(arduino)
END
```

Associated

`SCLOSE`, `SGET`, `SGET$`, `SOPEN`, `SPUT`, `SREADY`

SQRT

Purpose

Return the square root of the specified number.

Syntax

`squareroot=SQRT(number)`

Description

Returns the square root of the argument *number*. This is the opposite of multiplying a number by itself i.e. $X = \text{SQRT}(X * X)$

Example

```
foursquared=4*4
PRINT SQRT(foursquared)
END
```


SREADY

Purpose

Get the number of characters available to be read on an open serial port.

Syntax

count=SREADY(*handle*)

Description

Returns the number of characters available to be read from an open serial port. This can be used to poll the device to avoid stalling your program when there is no data available to be read.

Example

```
REM Read a character from a serial port
arduino=SOPEN("/dev/ttyUSB0", 115200)
IF SREADY(arduino) THEN
    char$=SGET$(arduino)
    PRINT char$
ENDIF
SCLOSE(arduino)
END
```

Associated

SCLOSE, SGET, SGET\$, SOPEN, SPUT, SPUT\$



STOP

Purpose

Stop a running program.

Syntax

STOP

Description

Program execution is stopped with a message indicating the current line number.

Example

```
INPUT "Enter the Password: ", pass$
IF pass$ <> "wibble" THEN
    PRINT "Password Incorrect"
    STOP
ENDIF
PRINT "Password Correct"
END
```

Associated

CONT



STOPCHAN

Purpose

Stop the playing of a sound sample.

Syntax

STOPCHAN(*handle*)

Description

This function stops the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE*. It cannot be resumed once stopped.

Example

```
channel=0
volume=70
SETCHANVOL(channel,volume)
intro=LOADSAMPLE("pacman_intro.wav")
PLAYSAMPLE(intro,channel,0)
WAIT(3)
STOPCHAN(intro)
END
```

Associated

[LOADSAMPLE](#), [PAUSECHAN](#), [PLAYSAMPLE](#), [RESUMECHAN](#), [SETCHANVOL](#)

STOPMUSIC

Purpose

Stop music playing completely.

Syntax

STOPMUSIC

Description

Stops a playing music track which cannot then be resumed.

Example

```
handle=LOADMUSIC("takeoff.wav")
SETMUSICVOL(70)
PLAYMUSIC(handle,1)
STOPMUSIC
```

Associated

[LOADMUSIC](#), [PAUSEMUSIC](#), [RESUMEMUSIC](#), [SETMUSICVOL](#)

STR\$

Purpose

Returns a string version of the supplied number.

Syntax

string\$=STR\$(*number*)

Description

Returns a string in the decimal (base 10) representation of *number*. This is useful if you want to append a number to a string. This is the opposite of the VAL function.

Example

```
PRINT "The Answer is "+STR$(42)
END
```

Associated

VAL

SWAP

Purpose

Swap the value of two variables.

Syntax

SWAP(*value1,value2*)

Description

This swaps the value of the 2 variables round. Both arguments must be the same type - i.e. Both numeric or both string.

Example

```
lowest=99
highest=0
IF lowest>highest THEN
    SWAP(highest,lowest)
ENDIF
PRINT "Lowest "; lowest
PRINT "Highest ";highest
END
```

SWITCH

Purpose

Test a value against many different values and execute different code.

Syntax

SWITCH (*variable*)

 { CASE *value* {, *value* }

commands

 ENDCASE }

 [DEFAULT

commands

 ENDCASE]

ENDSWITCH

Description

Simplify the writing of multiple IF. . . THEN. . . ELSE statements.

Rules:

- Every SWITCH must have a matching ENDSWITCH.
- Every CASE or DEFAULT statement must have a matching ENDCASE.
- Statements after a CASE statement must not run-into another CASE.
- The constants after the CASE statement (and the expression in the SWITCH statement) can be either numbers or strings, but you can't mix both.

Example

INPUT a

SWITCH(a)

 CASE 1,2

 PRINT "You entered 1 or 2"

 ENDCASE

 CASE 7

 PRINT "You entered 7"

 ENDCASE

 DEFAULT

 PRINT "You entered something else"

 ENDCASE

ENDSWITCH

END

Associated

[ENDCASE](#), [CASE](#), [ENDSWITCH](#)

TAN

Purpose

Return the tangent of the given angle.

Syntax

tangent=TAN(*angle*)

Description

Returns the tangent of the *angle* in the current angle units. In a right-angled triangle the tangent of an angle is the ratio of the length of the opposite side to the length of the adjacent side. This is a measure of the steepness of an angle.

Example

```
DEG
PRINT "Tangent of 45 degrees: "; TAN(45)
PRINT "ArcTangent of 1: "; ATAN(1)
END
```

Associated

ACOS, ASIN, ATAN, COS

TANGLE

Purpose

Read or set the current angle of the turtle.

Syntax

angle=TANGLE

TANGLE=*angle*

Description

This can be read or assigned to and represents the current angle of the turtle when using turtle graphics mode (in the current angle units)

Example

```
CLS
ORIGIN(GWIDTH/2,GHEIGHT/2)
CLOCK
FOR I = 1 TO 60 LOOP
    TANGLE = I
    MOVETO(0,0)
    PENDOWN
    MOVE(100)
REPEAT
END
```

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT

THEIGHT

Purpose

The height in characters of the display.

Syntax

height=THEIGHT

Description

The height in characters of the display.

Example

```
CLS
text$="This text is centred in the screen"
HVTAB((TWIDTH-LEN(text$))/2,THEIGHT/2)
PRINT text$
END
```

Associated

TWIDTH

TIME

Purpose

Find out how long the program has been running in 1000s of a second.

Syntax

time=TIME

Description

This returns a number which represents the time that your program has been running in milliseconds.

Example

```
REM Simple reaction timer
WAIT(2)
REM Make sure no key pressed
WHILE INKEY<>-1 LOOP
REPEAT
stime=TIME
PRINT "Go!"
WHILE INKEY=-1 LOOP
REPEAT
etime = TIME
PRINT "Your reaction time is ";
PRINT etime-stime; " milliseconds"
END
```

TIME\$

Purpose

Returns a string with the current time.

Syntax

```
now$=TIME$
```

Description

This returns a string with the current time in the following format: HH:MM:SS. For example: 18:05:45.

Example

```
PRINT "The time now is ";  
PRINT TIME$  
END
```

Associated

DATE\$

TONE

Purpose

Play a tone with the specified parameters.

Syntax

```
TONE(channel,volume,frequency,duration)
```

Description

This plays a simple tone of the given *frequency* (1 to 5000Hz), *volume* (%) and *duration* (0.01 to 20 seconds) on the given *channel*.

You can play multiple tones by playing them one after the other and up to three tones can be played simultaneously on different channels.

Channel 0 is white noise and the frequency has no bearing on the sound produced.

Example

```
Channel=1  
Volume=70  
FOR Note=1 TO 5 LOOP  
    READ Frequency,Duration  
    TONE(Channel,Volume,Frequency,Duration)  
REPEAT  
END  
DATA 440, 1, 493, 1, 392, 1, 196, 1, 294 ,2
```

Associated

SOUND

TRIANGLE

Purpose

Draw a triangle on the screen.

Syntax

TRIANGLE (*xpos1*, *ypos1*, *xpos2*, *ypos2*,
xpos3, *ypos3*, *fill*)

Description

Draws a triangle with its corners at the three given points. The final parameter, *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Example

```
LOOP
  COLOUR=RND(16)
  x1=RND(GWIDTH)
  x2=RND(GWIDTH)
  x3=RND(GWIDTH)
  y1=RND(GHEIGHT)
  y2=RND(GHEIGHT)
  y3=RND(GHEIGHT)
  f=RND(2)
  TRIANGLE (x1,y1,x2,y2,x3,y3,f)
  UPDATE
  IF INKEY<>-1 THEN BREAK
REPEAT
END
```

Associated

CIRCLE, ELLIPSE, RECT



TRUE

Purpose

Represents the logical "true" value.

Syntax

TRUE

Description

Represents a Boolean value that succeeds a conditional test. It is equivalent to a numeric value of 1 (in fact anything other than 0 evaluates to TRUE)

Example

```
condition=TRUE
IF condition=TRUE THEN
  PRINT "Condition is TRUE"
ENDIF
PRINT condition
END
```

Associated

FALSE



TWIDTH

Purpose

The width in characters of the display.

Syntax

width=TWIDTH

Description

The width in characters of the display.

Example

```
text$="This text is centred horizontally"
HTAB=(TWIDTH-LEN(text$))/2
PRINT text$
END
```

Associated

THEIGHT



UPDATEMODE

Purpose

Set the video update mode.

Syntax

UPDATEMODE=*mode*

Description (overleaf)



UPDATEMODE Description

The updateMode determines when the screen is redrawn. Redrawing the screen takes a little time and will slow down a program if you do it too often. The value of the *mode* parameter can be **0**, **1**, or **2** as follows:

0- automatic updates do not happen. Nothing will be drawn on the screen until the UPDATE command is issued.

1-This is the default mode whereby an UPDATE happens automatically when you output a new line, or the screen scrolls.

2-The screen is updated after every PRINT instruction whether it takes a new line or not.

Example

```
CLS
INPUT "Update Mode? ",mode
IF mode>=0 AND mode<=2 THEN
    PRINT "Press space to exit"
    WAIT(1)
    UPDATEMODE=mode
    LOOP
        PRINT "Hello World ";
    REPEAT UNTIL INKEY=32
    UPDATE
ELSE
    PRINT "Invalid Update Mode"
ENDIF
WAIT(1)
END
```

Associated

UPDATE

UNTIL REPEAT

Purpose

Loop until the specified condition is met.

Syntax

```
UNTIL condition LOOP
    {statements}
REPEAT
```

Description

Execute the *statements* zero or more times until the *condition* is TRUE (Not 0).

Because the test is done at the start of the loop the *statements* may not be executed at all

Example

```
REM Print 1 to 10
count=1
UNTIL count>10 LOOP
    PRINT count
    count=count + 1
REPEAT
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT,
REPEAT UNTIL, WHILE REPEAT



UPDATE

Purpose

Update screen graphics.

Syntax

```
UPDATE
```

Description

Graphics are drawn to a temporary screen buffer rather than the visible screen. The UPDATE command copies the working area to the main display. An update is also performed if your program stops for input, or when you PRINT a new line.

Example

```
REM Moire patterns
LOOP
    CLS
    COLOUR=RND(15)+1
    x=RND(GWIDTH)
    y=RND(GHEIGHT)
    FOR w=0 TO GWIDTH-1 STEP 3 LOOP
        LINE(x,y,w,0)
        LINE(x,y,w,GHEIGHT-1)
    REPEAT
    FOR h=0 TO GHEIGHT-1 STEP 3 LOOP
        LINE(x,y,0,h)
        LINE(x,y,GWIDTH-1,h)
    REPEAT
    UPDATE
REPEAT
END
```

Associated

UPDATEMODE



VAL

Purpose

Returns the number represented by a character string.

Syntax

```
number=VAL(string$)
```

Description

Returns the *number* represented by *string\$*. This is the opposite of the STR\$ function.

Example

```
now$ =TIME$
hh=VAL(LEFT$(now$, 2))
mm=VAL(MID$(now$, 3, 2))
ss=VAL(RIGHT$(now$, 2))
elapsed=hh*3600+mm*60+ss
PRINT "Seconds since midnight: "; elapsed
END
```

Associated

STR\$

VLINE

Purpose

Draws a vertical line.

Syntax

```
VLINE (ypos1,ypos2,xpos)
```

Description

Draws a vertical line on column *xpos*, from row *ypos1* to row *ypos2*.

Example

```
CLS
COLOUR=red
FOR xpos=0 TO GWIDTH STEP 100 LOOP
    VLINE(0, GHEIGHT, xpos)
REPEAT
UPDATE
END
```

Associated

HLINE, LINE, LINETO

VTAB

Purpose

Set/Read the current text cursor vertical position.

Syntax

VTAB=*value*

value=VTAB

Description

Set/Read the current text cursor vertical position.

Example

CLS

FOR ypos = 0 TO THEIGHT LOOP

 VTAB=ypos

 PRINT VTAB

REPEAT

VTAB=0

END

Associated

[HTAB](#), [HVTAB](#)

WAIT

Purpose

Waits for the specified time to elapse.

Syntax

WAIT(*time*)

Description

This waits (does nothing) for *time* seconds. This may be a rational number, but the accuracy will depend on the computer you are running it on, however delays down to 1/100th of a second should be achievable.

Example

REM COUNT 10 Seconds

CLS

Seconds = 0

FOR I=1 TO 10 LOOP

 WAIT(1)

 Seconds=Seconds + 1

 HVTAB(10,10)

 PRINT Seconds

REPEAT

PRINT "Elapsed "; TIME/1000

END

WHILE REPEAT

Purpose

Loop while the specified condition is met.

Syntax

```
WHILE condition LOOP
    {statements}
REPEAT
```

Description

Execute the *statements* zero or more times while the *condition* is TRUE (Not 0). Because the test is done at the start of the loop the *statements* may not be executed at all.

Example

```
handle=OPEN("whiletest.txt")
FOR r=0 TO 10 LOOP
    PRINT# handle,"Record ";r
REPEAT
CLOSE(handle)
handle = OPEN("whiletest.txt")
WHILE NOT EOF(handle) LOOP
    INPUT# handle, record$
    PRINT record$
REPEAT
CLOSE (handle)
END
```

Associated

[BREAK](#), [CONTINUE](#), [LOOP](#), [LOOP REPEAT](#), [FOR REPEAT](#),
[REPEAT UNTIL](#), [UNTIL REPEAT](#)



listGamepads

Immediate mode only. Displays a list of connected devices.

numButtons(gamepad)

Returns the number of buttons on a specified device

numAxes(gamepad)

Returns the number of analog axis on a specified device

numHats(gamepad)

Returns the number of HATS on a specified device

numGamepads

Returns the number of devices connected

getAxis(gamepad, axis)

Returns the value of a given axis on a specified device

getButton(gamepad, button)

Returns the value of a given button on a specified device

getHat(gamepad, hat)

Returns the value of a given HAT on a specified device

SetupGamepad(gamepad)

Configure a specified device

scanBackspace	scan1	scanA	scanV	scanDown	scanCapsLock
scanTab	scan2	scanB	scanW	scanRight	scanScrolLock
scanClear	scan3	scanC	scanX	scanLeft	scanRShift
scanReturn	scan4	scanD	scanY	scanInsert	scanLShift
scanPause	scan5	scanE	ScanZ	scanHome	scanRCtrl
scanEscape	scan6	scanF	scanDelete	scanEnd	scanLCtrl
scanSpace	scan7	scanG	scanKP0	scanPageup	scanRAlt
scanExclaim	scan8	scanH	scanKP1	scanPagedown	scanLAlt
scanQuoteDbl	scan9	scanI	scanKP2	scanF1	scanRMeta
scanHash	scanColon	scanJ	scanKP3	scanF2	scanLMeta
scanDollar	scanSemiColon	scanK	scanKP4	scanF3	scanLSuper
scanAmpersand	ScanLess	scanL	scanKP5	scanF4	scanRSuper
scanQuote	ScanEquals	scanM	scanKP6	scanF5	scanMode
scanLeftParen	scanGreater	scanN	scanKP7	scanF6	scanCompose
scanRightParen	scanQuestion	scanO	scanKP8	scanF7	scanHelp
scanAsterisk	scanAt	scanP	scanKP9	ScanF8	scanPrint
scanPlus	scanLeftBracket	scanQ	scanKpPeriod	scanF9	scanSysReq
scanComma	scanBackSlash	scanR	scanKpDivide	scanF10	scanBreak
scanMinus	scanRightBracket	scanS	scanKpMultiply	scanF11	scanMenu
scanPeriod	scanCaret	scanT	scanKpMinus	scanF12	scanPower
scanSlash	ScanUnderscore	scanU	scanKpPlus	scanF13	scanEuro
scan0	scanBackQuote	scanV	ScanKpEnter	scanF14	scanUndo
			ScanKpEquals	ScanF15	
			scanUp	scanNumLock	

Note	Octave Number						
	1	2	3	4	5	6	7
B	0	49	97	145	193	241	
A#	0	45	93	141	189	237	
A		41	89	137	185	233	
G#		37	85	133	181	229	
G		33	81	129	177	225	
F#		29	77	125	173	221	
F		25	73	121	169	217	
E		21	69	117	165	213	
D#		17	65	113	161	209	
D		13	61	109	157	205	253
C#		9	57	105	153	201	249
C		5	53	101	149	197	245



FUZE BASIC

Programmer's reference guide

£14.99 / \$24.99 / €21.99



0 799439 183735 >

FUZE BASIC
Programmer's Reference Guide

fuze.co.uk